神经网络与深度学习应用实战

刘凡平 等编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

VISIT…

内 容 简 介

本书结合实际应用介绍神经网络和深度学习等技术领域相关信息。从结构上重点介绍了前馈型神经网络、反馈型神经网络，以及自组织竞争型神经网络，并针对当下深度学习中比较重要的网络进行了详细介绍，包括卷积神经网络、循环（递归）神经网络、深度信念网络、生成对抗网络，以及深度强化学习。本书不仅能让读者对当前神经网络和深度学习技术有体系的认知，更能让读者在人工智能领域进行一些深入思考。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

神经网络与深度学习应用实战 / 刘凡平等编著. —北京：电子工业出版社，2018.3
ISBN 978-7-121-33718-5

Ⅰ. ①神… Ⅱ. ①刘… Ⅲ. ①人工神经网络－应用－研究②机器学习－应用－研究 Ⅳ. ①TP183 ②TP181

中国版本图书馆 CIP 数据核字(2018)第 031212 号

责任编辑：安　娜
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　印张：15.75　字数：300 千字
版　　次：2018 年 3 月第 1 版
印　　次：2018 年 3 月第 1 次印刷
定　　价：69.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

前言

本书结合实际应用介绍神经网络和深度学习等技术领域相关信息，从结构上重点介绍了前馈型神经网络、反馈型神经网络，以及自组织竞争型神经网络，并针对当下深度学习中比较重要的网络进行了详细介绍，包括卷积神经网络、循环（递归）神经网络、深度信念网络、生成对抗网络，以及深度强化学习。本书不仅能让读者对当前神经网络和深度学习技术有体系的认知，更能让读者在人工智能领域进行一些深入思考。

读者对象

- 对神经网络、深度学习以及人工智能有兴趣的读者；
- 对算法以及机器学习领域有兴趣的读者；
- 互联网行业不同层次的从业者；
- 软件工程或计算机相关专业的在校学生。

本书特色

书中内容紧密结合当前一线工程师工作研究成果，是对当前神经网络和深度学习的完整性原理介绍和实践分析。本书充分利用了最新技术发展的应用成果，不仅结合原理分析，还结合案例进行辅助理解。

本书介绍的相关深度学习技术广泛应用于各个领域，可以在自然语言处理、计算机视觉、文本分析等领域中应用，在当前甚至未来三到五年，都具有实际意义。

本书结构

本书按照由浅入深、循序渐进的顺序对神经网络和深度学习的内容进行介绍。全书共分为三篇，分别从基础、进阶、高阶三个层次逐步展开，总共 12 章，各章的主要内容如下。

第 1 章阐述了在当前时代背景下，神经网络和人工智能的发展历程，针对未来人工智能极

可能改变的领域进行了深入介绍，并介绍了深度学习与机器学习的关系，以及深度学习与人工智能的关系。

第 2 章介绍了神经网络和深度学习的数学基础，从向量、矩阵、导数、数值计算、概率分布、参数估计等方面进行了详细介绍，为学习后续内容奠定基础。

第 3 章重点介绍了机器学习的基础内容，神经网络和深度学习都属于机器学习中的内容，包括拟合问题、交叉检验、产生式与判别式模型等，有助于加强对神经网络和深度学习的理解。

第 4 章介绍了神经网络的基础，包括神经网络中常见的学习方法以及神经网络的优化方法，阐述了常见的神经网络类型以及深度学习中的网络类型，并介绍了深度学习与多层神经网络的关系。

第 5 章重点介绍了前馈型神经网络，它是神经网络中极为重要的一种网络类型。本章从单层感知器开始，逐步深入介绍了 BP 神经网络以及径向基函数神经网络，重点介绍了反向传播算法。

第 6 章详细介绍了反馈型神经网络，它是一种带联想记忆的神经网络。本章重点介绍了 Hopfiled 神经网络、Elman 神经网络以及递归神经网络。对于递归神经网络还进行了更为深入的介绍，包括其反向传播算法以及各类改进的结构。

第 7 章重点介绍了自组织竞争型神经网络，从传统的系统聚类法、基于划分的聚类算法、基于密度的聚类算法、基于层次的聚类算法开始，详细介绍了自组织竞争型神经网络中的典型代表——自组织映射网络，还介绍了自适应共振理论以及对偶传播网络。

第 8 章介绍了卷积神经网络，卷积神经网络是目前图像处理中比较优秀的神经网络。本章重点介绍了卷积神经网络中的卷积、卷积核等重要基础概念，详细阐述了卷积神经网络中各层的工作原理，并介绍了常见的间距神经网络结构。

第 9 章介绍了循环神经网络，循环神经网络与递归神经网络有一定的相似性。本章介绍了一般的循环神经网络，包括单向循环神经网络、双向循环神经网络以及深度循环神经网络。重点介绍了长短时记忆网络。

第 10 章介绍了深度信念网络，深度信念网络是由受限玻尔兹曼机组成的网络结构。本章重点介绍了受限玻尔兹曼机的逻辑结构和工作原理，并介绍了深度信念网络的训练过程。

第 11 章介绍了生成对抗网络，生成对抗网络是未来会有较大突破的网络结构之一。本章从一般的生成对抗网络入手进行介绍，然后介绍了各类改进版本，包括 DCGAN、CGAN、WGAN 等，并对生成对抗网络的未来做了一定猜想。

第 12 章介绍了深度强化学习，深度强化学习是一种有别于传统的有监督学习和无监督学习的学习方式。本章重点介绍了强化学习的工作原理、马尔科夫决策过程等，并结合强化学习的各类算法进行了详细的介绍。

上述章节中，郭武彪完成了第 6 章内容的编写，陈相礼完成了第 9 章内容的编写，杨华完成了第 11 章内容的编写以及本书格式校验，其余章节内容由刘凡平完成编写，并对本书内容进行了校验。

由于时间仓促及编者水平有限，书中难免存在错误和不足之处，恳请广大读者多多理解，并批评指正，也可以通过邮箱（liufanping@iveely.com）联系我们。

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- ◎ **提交勘误**：您对书中内容的修改意见可在 提交勘误 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- ◎ **交流互动**：在页面下方 读者评论 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/33718

致谢

本书的内容基本都来自我们的工作经验，感谢曹杨、黄诚，正是你们的支持和鼓励，以及工作中的栽培和肯定，才能使得我们（陈相礼、郭武彪、杨华、刘凡平）有能力和勇气写出这本书。还有那些曾经一起学习和共事的朋友，你们给予我们很多无私的帮助，使得我们在和你们相处的过程中能够快速地成长，感谢一路上有你们的陪伴，正是有了你们，沿途的风景才格外的美丽。

衷心感谢我们的家人，感谢你们在过去的时间里对我们的理解和支持，为我们营造了一个良好的写作环境，并鼓励我们坚持认真写作，使得本书能够顺利完成。

本书编写过程中还得到了很多朋友的支持和帮助，限于篇幅，虽然不能一一对你们表示感谢，但是对你们同样心怀感激。

最后，感谢这个时代，给予每一个有理想的人，赋予实现人生价值的机会！

目录

基础篇

第 1 章　时代崛起 2

1.1　概要 2

1.1.1　基本概念 2

1.1.2　深度学习与机器学习的关系 4

1.1.3　深度学习与人工智能的关系 5

1.2　历史发展 5

1.2.1　神经网络发展历史 5

1.2.2　人工智能发展历史 7

1.3　应用领域 8

1.3.1　智能个人助理 8

1.3.2　智能安防 9

1.3.3　无人驾驶 9

1.3.4　电商零售 11

1.3.5　智慧医疗 11

1.3.6　金融服务 12

1.3.7　智能教育 13

1.4　未来猜想 14

1.4.1　人文的快速发展 14

1.4.2　人类也是“机器人” 14

1.4.3　新的不平等现象 15

1.5　本章小结 16

第 2 章 数学理论基础.....17
2.1 向量.....17
2.1.1 相关概念.....17
2.1.2 向量的线性相关性.....18
2.1.3 向量的外积.....18
2.1.4 向量夹角与余弦相似性.....18
2.1.5 实例：基于向量夹角的文本相似性分析.....19
2.2 矩阵.....20
2.2.1 矩阵乘法.....20
2.2.2 克罗内克积.....21
2.3 导数.....22
2.3.1 概述.....22
2.3.2 一般运算法则.....22
2.3.3 链式求导法则.....23
2.4 数值计算.....23
2.4.1 误差.....23
2.4.2 距离.....24
2.4.3 数值归一化.....26
2.5 概率分布.....26
2.5.1 二项分布.....26
2.5.2 超几何分布.....27
2.5.3 泊松分布.....27
2.5.4 指数分布.....28
2.5.5 正态分布.....29
2.6 参数估计.....29
2.6.1 概率.....29
2.6.2 贝叶斯估计.....30
2.6.3 最大似然估计.....31
2.6.4 最大后验估计.....32
2.7 回归分析.....33
2.7.1 线性回归.....33
2.7.2 逻辑回归.....36

2.8 判定问题 39
2.8.1 P 问题 39
2.8.2 NP 问题 39
2.8.3 NP-Complete 问题 40
2.8.4 NP-Hard 问题 40
2.9 本章小结 41

第 3 章 机器学习概要 42

3.1 机器学习的类型 42
3.1.1 有监督学习 42
3.1.2 无监督学习 43
3.1.3 强化学习 43
3.2 机器学习中常见的函数 44
3.2.1 激活函数 44
3.2.2 损失函数 47
3.2.3 核函数 48
3.3 机器学习中的重要参数 49
3.3.1 学习速率 49
3.3.2 动量系数 50
3.3.3 偏置项 50
3.4 拟合问题 51
3.4.1 过拟合现象 51
3.4.2 欠拟合现象 52
3.4.3 解决过拟合问题的一般方法 52
3.4.4 实例：拟合与二元一次方程求解 55
3.5 交叉检验 55
3.5.1 数据类型种类 55
3.5.2 留一交叉验证 57
3.5.3 K 折交叉验证 57
3.6 线性可分与不可分 58
3.7 机器学习的学习特征 59
3.8 产生式模型与判别式模型 60
3.9 机器学习效果的一般评价指标 61

3.10 本章小结 63

第 4 章 神经网络基础 64

4.1 概述 64
4.1.1 神经网络模型 64
4.1.2 经典的神经网络结构 65
4.1.3 一般业务场景中神经网络适应性 66
4.1.4 神经网络的深度 67
4.2 常见学习方法 67
4.2.1 误差修正学习 67
4.2.2 赫布学习规则 68
4.2.3 最小均方规则 69
4.2.4 竞争学习规则 70
4.2.5 其他学习规则 71
4.3 优化方法：梯度下降 72
4.3.1 概述 72
4.3.2 梯度下降法 72
4.3.3 梯度下降的优化算法 74
4.3.4 梯度消失问题 76
4.3.5 示例：利用梯度下降法求函数极值 77
4.4 常见的神经网络类型 78
4.4.1 前馈型神经网络 78
4.4.2 反馈型神经网络 79
4.4.3 自组织竞争型神经网络 79
4.5 深度学习中常见的网络类型 80
4.5.1 卷积神经网络 80
4.5.2 循环神经网络 80
4.5.3 深度信念网络 80
4.5.4 生成对抗网络 81
4.5.5 深度强化学习 81
4.6 其他神经网络与深度学习 82
4.6.1 随机神经网络 82
4.6.2 量子神经网络 82

4.6.3 迁移学习 82
4.7 深度学习与多层神经网络的关系 83
4.8 调参技巧 84
4.9 本章小结 85

进阶篇

第 5 章 前馈型神经网络 88
5.1 概述 88
5.2 常见结构 88
5.3 单层感知器网络 89
5.3.1 原理 89
5.3.2 网络结构 90
5.3.3 实例一：基于单层感知器“与”运算 90
5.3.4 实例二：利用感知器判定零件是否合格 91
5.4 BP 神经网络 93
5.4.1 概述 93
5.4.2 反向传播算法 93
5.4.3 异或问题的解决 96
5.4.4 避免病态结果 98
5.4.5 实例：基于多层感知器的手写体数字识别 99
5.5 径向基函数神经网络 101
5.5.1 原理介绍 101
5.5.2 中心选择方法 102
5.5.3 训练过程 103
5.5.4 径向基函数神经网络与 BP 神经网络的差异 104
5.6 本章小结 105

第 6 章 反馈型神经网络 107
6.1 概述 107
6.1.1 基本原理 107
6.1.2 与前馈型神经网络的差异 108
6.2 Hopfield 神经网络 109

6.3 Elman 神经网络 112
6.3.1 结构组成 112
6.3.2 学习算法 112
6.4 递归神经网络 113
6.4.1 产生背景 114
6.4.2 基本结构 115
6.4.3 前向计算过程 116
6.4.4 反向传播：BPTS 算法 117
6.4.5 应用场景 118
6.4.6 递归神经网络的结构改进 118
6.4.7 应用实例 121
6.5 本章小结 124

第 7 章 自组织竞争型神经网络 125

7.1 概述 125
7.1.1 一般网络模型 125
7.1.2 工作原理 126
7.1.3 实例：用竞争学习规则进行模式分类 127
7.2 常见的聚类方法 129
7.2.1 系统聚类法 129
7.2.2 基于划分的聚类算法 130
7.2.3 基于密度的聚类算法 131
7.2.4 基于层次的聚类算法 132
7.3 自组织映射网络 134
7.3.1 概述 134
7.3.2 训练算法 134
7.3.3 实例：利用自组织映射网络划分城市群 135
7.3.4 优劣势分析 136
7.4 其他自组织竞争型神经网络 137
7.4.1 自适应共振理论 137
7.4.2 对偶传播神经网络 138
7.5 本章小结 139

高阶篇

第 8 章　卷积神经网络 142

8.1　概述 142

8.1.1　发展背景 142

8.1.2　基本概念 143

8.1.3　基本网络结构 144

8.2　卷积 145

8.2.1　卷积的物理意义 145

8.2.2　卷积的理解 145

8.2.3　卷积的实例 147

8.3　卷积核 148

8.3.1　卷积核的含义 148

8.3.2　卷积操作 150

8.3.3　卷积核的特征 150

8.4　卷积神经网络中各层工作原理 151

8.4.1　卷积层 151

8.4.2　下采样层 151

8.4.3　Softmax 层 152

8.5　卷积神经网络的逆向过程 153

8.6　常见卷积神经网络结构 154

8.6.1　LeNet-5 154

8.6.2　AlexNet 155

8.7　应用场景与效果评估 157

8.7.1　场景 1：图像分类 157

8.7.2　场景 2：目标检测 158

8.7.3　场景 3：实例分割 159

8.8　Maxout Networks 160

8.9　本章小结 162

第 9 章　循环神经网络 163

9.1　概述 163

9.2　一般循环神经网络 164

9.2.1 概述 164
9.2.2 单向循环神经网络 165
9.2.3 双向循环神经网络 166
9.2.4 深度循环神经网络 167
9.3 训练算法：BPTT 算法 168
9.3.1 前向计算 168
9.3.2 误差项计算 169
9.3.3 权值梯度计算 169
9.3.4 梯度爆炸与梯度消失问题 170
9.4 长短时记忆网络 170
9.4.1 背景 170
9.4.2 核心思想 171
9.4.3 详细结构 172
9.4.4 训练过程 176
9.4.5 相关变种简介 181
9.5 常见循环神经网络结构 182
9.5.1 N 比 N 结构 182
9.5.2 N 比 1 结构 183
9.5.3 1 比 N 结构 183
9.5.4 N 比 M 结构 184
9.6 与自然语言处理结合 185
9.7 实例：文本自动生成 186
9.8 本章小结 187

第 10 章 深度信念网络 188

10.1 概要 188
10.1.1 背景 188
10.1.2 基本结构 188
10.2 受限玻尔兹曼机 190
10.2.1 概述 190
10.2.2 逻辑结构 192
10.2.3 对比分歧算法 194
10.3 训练过程 194

10.3.1 工作流程 194
10.3.2 调优过程 195
10.4 本章小结 196

第 11 章 生成对抗网络 197

11.1 概述 197
11.1.1 背景概要 197
11.1.2 核心思想 198
11.1.3 基本工作流程 199
11.2 朴素生成对抗网络 201
11.2.1 网络结构 201
11.2.2 实例：基于朴素生成对抗网络生成手写体数字 203
11.3 深度卷积生成对抗网络 206
11.3.1 产生背景 206
11.3.2 模型改进 206
11.3.3 网络结构 207
11.3.4 实例：基于深度卷积对抗网络生成手写体数字 208
11.4 条件生成对抗网络 212
11.4.1 网络结构 212
11.4.2 实例：CGAN 结合 DCGAN 生成手写体数字 213
11.5 瓦瑟斯坦生成对抗网络 214
11.5.1 概述 214
11.5.2 差异化 215
11.5.3 实例：WGAN 结合 DCGAN 生成手写体数字 216
11.6 生成对抗网络的探索 217
11.6.1 价值与意义 217
11.6.2 面临的问题 218
11.6.3 应用场景示例 218
11.6.4 未来探索 220
11.7 本章小结 220

第 12 章 深度强化学习 221
12.1 概述 221
12.1.1 概要 221
12.1.2 基本原理 222
12.2 马尔科夫决策过程 223
12.2.1 马尔科夫过程 223
12.2.2 隐马尔科夫模型 224
12.2.3 马尔科夫决策过程 225
12.3 深度强化学习算法 229
12.3.1 DQN 算法 229
12.3.2 A3C 算法 231
12.3.3 UNREAL 算法 231
12.4 强化学习的探索 232
12.4.1 应用场景探索 232
12.4.2 面临的问题 233
12.5 本章小结 234

基础篇

第 1 章

时代崛起

每个时代，都有每个时代的主题，对于当前而言，所谓的互联网时代、移动互联网时代等都为人工智能时代作了铺垫，每一个智能设备背后的用户，都在为这个时代积极奉献。

1.1 概要

大数据人工智能技术，在应用层面包括机器学习、神经网络、深度学习等，它们都是现代人工智能的核心技术。在大数据背景下，这些技术均得到了质的提升，人工智能、机器学习和深度学习的包含关系如图 1-1 所示。

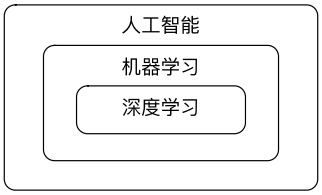


图 1-1　人工智能、机器学习和深度学习的关系

1.1.1 基本概念

1. 机器学习

机器学习（Machine Learning）也被称为统计学习理论，是人工智能的重要分支。它通过数据分析获得数据规律，并将这些规律应用于预测或判定其他未知数据。机器学习目前已经广泛应用于数据挖掘、自然语言处理、语音识别等，尤其是在搜索引擎领域。

搜索引擎是人工智能技术发展的先锋队，目前百度已经定位为一家人工智能公司，同时搜狗王小川也将人工智能视为未来。在海量数据面前，机器学习的方法成效显著，具体算法包括决策树、感知器、支持向量机、马尔科夫链、最近邻居法等。拥有大规模用户的搜索引擎业务

的公司是最先接触到大数据的企业，它们对于机器学习的需求远远超过其他公司。这类公司利用人工智能技术的原因是希望其搜索结果更加精准，甚至能直接命中用户答案。

人工智能的发展经历了从“推理”到“知识”、从“知识”到“学习”的重要过程，机器学习一直在人工智能的道路上解决问题。机器学习不是一个单一的学科，而是与数学、计算机、生物学等多领域有交叉的学科。机器学习目前不仅应用在搜索引擎中，在生物特征识别、生物医学研究、证券分析等里领域都有深入应用，并取得了不错的成绩。

从另外一个角度看待机器学习，机器学习的“学习”意味着机器学习的算法尝试沿着某个维度进行优化，可以理解为它们通常尝试以最小的错误率来最大限度地发挥其预测的可能性。因此产生了三个名称：错误函数、损失函数以及目标函数，因为每一个机器学习算法都有一个学习目标。

项目或工程中使用机器学习算法时，可以通过确定输入、输出以及目标函数和最小错误率来评估其算法的作用与效果。

对于机器学习算法中的输入和输出，通常情况下，初步测试的输入与输出的对应结果都是错误的，如果拥有与输入有关的输出结果对应关系，那么可以通过与期望的输出结果对比来衡量猜测的准确度，然后使用该错误来修改算法，这是有监督学习的常见方式。它们不断估算输出结果并修改估算过程的参数，直到错误率达到极值。

2. 神经网络

神经网络（Artificial Neural Network）是机器学习的一个重要算法，也是奠定深度学习发展的基础算法，它的思想影响了深度学习，使得深度学习成为人工智能中极为重要的技术之一。

神经网络作为一种常用的方法，是一种通过模仿生物的神经网络结构和功能的数学模型，也是一种自适应的计算模型。它通过感知外部信息的变化来改变系统的内部结构。神经网络由许多的神经元组成，神经元之间相互联系构成信息处理的庞大网络。假设做一件事情有多种途径，那么神经网络会告知设计者哪一种途径是最佳方式。

神经网络的优势在于它是一个能够通过现有数据进行自我学习、总结、归纳的系统，能够推理产生一个智能识别系统，从而成为人工智能技术中的重要基石。

3. 深度学习

深度学习（Deep Learning）是机器学习的重要分支，也是传统神经网络的重要延伸。深度学习的网络结构已有很多，例如深度神经网络、卷积神经网络、递归神经网络等。作为多层非线性神经网络模型，它拥有强大的学习能力，通过与大数据、云计算和 GPU 并行计算相结合，

它在图形图像、视觉、语音等方面均获得较好成就，远远超越了传统机器学习的效果，因此深度学习被大众视为人工智能前进的重要一步。2016 年 3 月，以深度学习为基础的人工智能围棋应用 AlphaGo 在围棋比赛中战胜人类围棋高手，成为热议话题。

深度学习目前在图像处理、语音识别、生物特征识别等领域中已经获得了广泛的应用，并得到行业较高评价，这使得深度学习持续发展，加速推进人工智能的发展。

1.1.2 深度学习与机器学习的关系

在以往，绝大多数的机器学习方式都是浅层结构，即使采用非线性处理的方式，这些浅层结构的深度往往也不会太深，这种状况在近十年左右才得以改变；随着计算能力的增强，计算的深度也在不断增加。

机器学习中常见的浅层结构包括高斯混合模型、支持向量机、最大熵模型、逻辑回归、多层感知器，等等。实践不断告诉我们，浅层结构在解决一些简单问题时效果比较明显，但是对于处理复杂多变的问题，例如语音、视频等则效果较差。

深度学习的基础研究源自神经网络。神经网络中最为常见的是前馈型神经网络，倘若具备多隐藏层则可以被称作深度神经网络（Deep Neural Network）。深度神经网络能够显著提升问题的处理效果，虽然目前训练过程中需要强大的计算能力，但是借助 GPU 以及分布式计算，可以在保障效果的前提下有效提升计算效率。

深度学习可以处理任何类型的数据，例如：

（1）声音。主要针对语音识别、语音合成、语音模拟等。

（2）文本。包含自然语言处理、自然语言生成等。

（3）图像。针对计算机视觉领域，包括图像分类、图像目标检测、图像语义分割等。

（4）时间序列。主要在数据传感、关联事件分析等细分领域。

（5）视频。主要在视频内容理解，智能视频广告等领域。

深度学习可以解决几乎任何机器感知的问题，包括对数据进行分类、聚类或对其进行预测分析。

（1）分类：例如对垃圾邮件和非垃圾邮件的归档处理。

（2）聚类：例如对相似性较高的文档进行归档处理。

（3）预测：例如根据历年的气象数据和最近的天气变化预测未来一周的天气情况。

深度学习非常适用于非结构化数据，例如上面提到的图像、视频、声音以及文本等。一个图像是像素的组合，一个消息是文字的组合。这些数据没有按行和列组织在典型的关系数据库中，这使得浅层结构的机器学习方式对其进行特征分析相对较为困难。深度学习的常用用例包括情感分析、图像分类、预测分析、推荐系统、异常检测等。

1.1.3　深度学习与人工智能的关系

从图 1-1 中，已知深度学习是机器学习的一个子集，而机器学习又是人工智能的一个子集，因此深度学习是人工智能的一个子技术分支。

智能程序是一种广泛应用的计算机智能程序，它可以通过一系列的条件判断形成，但是这样的智能程序往往很容易理解，因此智能程序不能视作当前的人工智能。人工智能是面向数据深入地分析结果，结果的推测过程不是人为可快速推测的，而是需要计算机辅助完成。深度学习则是借助计算机完成的、较好的人工智能技术。

既然机器学习是人工智能的技术分支，那么总有部分算法或模型属于人工智能领域但是不属于机器学习领域，例如规则引擎、专家系统、进化算法等，它们都属于人工智能的技术体系，但并不是机器学习。

深度学习是人工智能一个技术子集。深度神经网络在一系列重要领域，例如图像识别、声音识别、推荐系统等不断刷新各项指标，甚至超越了人类的认知范围。由 DeepMind 研发的著名人工智能程序 AlphaGo，在 2016 年击败了前世界围棋冠军李世乭，这也是深度学习技术对各领域影响的场景之一。

深度学习中的“深”是一个技术术语，一般而言，它指的是神经网络中的层数。一个浅层网络有一个所谓的隐藏层，而一个深层网络则有一个以上。对于一般简单的数据特征，它会从网络层的一层传递到下一层用映射关系表示，而深度神经网络的层次结构可以表达更为复杂的数据映射关系，以表示更复杂的特征。人工智能面向的问题也具备多层次数据的复杂特征，因此深度学习有效地解决了目前各行各业中的部分复杂问题。

1.2　历史发展

1.2.1　神经网络发展历史

神经网络作为深度学习的基础，它的发展历史影响了深度学习的发展，从而影响了人工智能的历史。

最初的神经元模型是由心理学家沃伦·麦卡洛克和数理逻辑学家沃尔特·皮茨在 1943 年提出的，他们在分析和总结人类神经元的基础上得到了该模型。最初的神经元模型奠定了后续神经网络的发展，包括 70 多年后的今天，众多的神经网络模型均是在最初的神经元模型的基础上发展而来的。

1956 年感知器的诞生给神经网络的发展带来了机会，感知器首次从理论研究走向工程实践，并取得了一定的效果。它由弗兰克·罗森布拉特提出，通过简单的加减法规则实现了两层的计算机学习网络。许多计算机实验室基于感知器的理论，从事计算机的文字识别、声音识别以及学习记忆问题等研究，但是此次的神经网络研究并未持续太久，主要有以下两方面原因。

（1）计算机处于全速发展时期。数字计算机在当时得到快速发展，研究者将重心转移到数字计算机，并认为数字计算机可以解决感知器可以解决的问题。

（2）电子工艺相对落后。当时的电子元件主要是基于电子管或晶体管，基于它们构建的神经网络成本较高、体积也较为庞大，用这样的电子工艺去模拟人的神经网络显得不够现实。

此外，后续的研究者也逐渐发现，当时的神经网络解决的问题基本是基于线性的，不足以解决一些复杂问题，甚至连计算机里的异或问题都不能有效解决，这使得不少神经网络的研究者认为神经网络的前景堪忧，甚至有不少研究者放弃已有研究。

在后来相当长的一段时间，感知器受到冷落，神经网络的发展也相对缓慢，直到 1975 年，神经网络中的反向传播算法的诞生，才有效解决了异或问题以及多层神经网络的训练问题。

20 世纪 80 年代初，超大规模集成电路的制作技术达到一个较高水平，实用化已经比较成熟，数字计算机的发展虽然得到了广泛的应用，但是在某些领域还是遇到了一些困难。同时，分布式并行处理也逐渐流行起来，正是在这样的背景下，物理学家 Hopfield 在 1982 年和 1984 年分别发表了关于神经网络的论文，引起了广泛的关注；1986 年大卫·鲁梅尔哈和詹姆斯·麦克莱伦对于分布式并行处理在计算机模拟神经活动的应用提供了全面的论述，使人们重新认识到神经网络的实用价值以及客观的可行性。因此在 20 世纪 80 年代中后期，又掀起了一股人工神经网络的研究热潮。

20 世纪 90 年代，神经网络的发展得到了持续发展，1990 年，罗德尼·布鲁克斯提出了用环境交互的方式打造 AI 机器人的设想；1997 年赛普·霍克赖特和于尔根·施密德胡伯提出长短期记忆神经网络，同年 IBM 研发的“深蓝”成为第一个打败人类国际象棋冠军的计算机程序；1998 年燕乐存和约书亚·本吉奥提出了关于利用神经网络进行手写识别和优化反向传播的方法。

近几年，神经网络的发展不再停留在传统神经网络的发展之上，而是深入研究神经网络能

够带来的更多效果，因此衍生了类似于卷积神经网络、递归神经网络等不同于传统神经网络的结构。

1.2.2　人工智能发展历史

近几年人工智能之所以能够快速发展，源于计算能力的提升和大数据提供的数据特征支持，大数据的发展也必然带动云计算能力的提升。曾经，很多人都说人工智能“已死”，一直处于停滞不前的状态，但是随着近些年大数据的发展，人工智能似乎是“死而复生”，获得了新的发展机遇。

人工智能技术是计算机科学发展的重要分支，它自 1956 年达特茅斯会议中诞生，它的第一次黄金时代是在 1956 年至 1974 年，这个阶段，计算机逐渐可以解决代数应用相关的问题，LISP 语言也在 1960 年诞生，是令人兴奋的一个开端。1974 年至 1980 年是人工智能发展的低谷期。

1980 年开始，人工智能技术似乎迎来了新的繁荣，但是仅仅七年之后，人工智能的冬天就又到来了。20 世纪 90 年代中期以后，神经网络研究取得新进展，以“深蓝”战胜国际象棋的世界冠军为里程碑，似乎又进入了新的人工智能时代。

近几年，人工智能也取得了不少的成果。2009 年，谷歌开始秘密研发无人汽车，2014 年，谷歌汽车在内华达州通过了自动驾驶汽车的测试。同样在 2009 年，美国西北大学智能信息实验室开发了 Stats Monkey，一个不依赖于人工参与的自动化体育新闻撰写程序；2010 年，计算机视觉领域的 ILSVCR 正式举办，到 2017 年最后一届，人工智能在计算机视觉的某些领域中已经超越人类。2011 年，在德国交通标志检测竞赛中，凭借基于卷积神经网络的体系结构，计算机的识别准确率达到 99.46%，而人类只有 99.22%。2016 年、2017 年谷歌 DeepMind 研发的 AlphaGo 超越世界的各顶尖围棋高手……这些都是近些年人工智能取得的显著的成效。

但是，上述这些只是让计算机感知和理解世界，而决策是人工智能领域的核心问题。计算机视觉和其他感知问题需要向计算机输入感知信息，计算机可以理解，并且要求计算机能够根据感知信息判断，输出正确的行为。为了使计算机做出良好的决策控制，就要求计算机具有一定的“思考”能力，使计算机能够学会掌握解决各种问题的能力，而这是强人工智能的研究目标。目前强人工智能还面临诸多问题：

（1）较弱的泛化能力。要加强通用的泛化能力，则必然会减少域知识（领域知识）的依赖性，而目前的大多数训练都需要非常专业的领域知识，以达到在专业领域的应用，但是同时它的泛化通用能力被降低。

（2）非通用的学习能力。无论是通过逻辑推理的演绎方法来学习，还是基于经验和记忆的

归纳能力，学习的方式都需要更加通用，目前大多数学习的模型都不能相互使用，一个场景一种学习方式。

（3）较弱的反思能力。目前的学习都是针对性的学习，反思能力是需要将学习的过程记录下来，将“学习的方法”也作为机器学习的一部分，即学习的方法也是可以学习的，通过自我对学习过程的反思，达到融会贯通的目的。

在 AlphaGo 出现之前，人们认为人工智能对人类生活的影响可能需要十多年的时间，但真实的技术发展速度已经超越普通大众的想象。

任何技术的发展都是充满崎岖的，人工智能则是典型的代表。从科学技术发展史上看，技术解决人类的需求是永无止境的，无论现在的人工智能发展到哪个层次，它对于人类未来的促进作用一定是潜移默化的，社会生活和人工智能在未来的发展中将会更加紧密相连。在当前大数据和云计算的时代，数据的多样性和丰富性更加显现，数学模型和算法也在不断改进，人工智能正在迎来它的第三次繁荣，未来充满了无限想象。

1.3 应用领域

在当前技术发展趋势下，人工智能（弱人工智能）已经不刻意去研究模拟人脑，而是利用大数据，通过计算机视觉、数据挖掘和统计理论，深化演绎、推理、归纳去解决现实中的各类问题。人工智能对未来的发展和帮助是不言而喻的，例如现实中的指纹识别、人脸识别、视网膜识别、声纹识别等，都是人工智能的实际应用案例。人工智能已经开始走向垂直化，从垂直领域而言，包括智能个人助理、无人驾驶、智能机器人、智慧医疗等。

1.3.1 智能个人助理

智能个人助理（Intelligent Personal Assistant）的主要落脚点在于智能手机的语音助手、语音输入、家庭管家以及陪伴机器人相关方面，类似的产品包括：微软小冰、百度度秘、Google Home、科大讯飞语音助手、Amazon Echo 等。

人工智能应用于个人助理，可以有效地了解用户的偏好和行为习惯，帮助用户进行日常管理、日程安排、问答服务等。从技术的角度，记录用户的行为和使用习惯可以对用户进行个性化特征和群体特征分析，利用云计算和人工智能分析技术，将非结构化的数据，例如电子邮件、图片、视频等数据进行结构化抽取或智能推理，并理解用户的语义和语境，从而实现良好的人机交互。

目前智能个人助理还处于初级阶段，主要的服务范围是信息检索以及帮助用户获得资讯信

息。无论是初创公司，还是规模较大的公司，在智能个人助理上的效果都还不能通过图灵测试，因此，目前的智能个人助理偏向于基础服务，例如天气预报、日程预定、购买机票、外卖预定等。智能个人助理的未来发展还需要借助大规模人工智能技术对服务进行升级，例如，增强基于上下文的对话能力，理解用户口语中的逻辑，增强服务的执行能力，情感感知能力等。随着各项技术的不断提升，智能个人助理的人机交互也将会更加自然流畅。

1.3.2　智能安防

安防行业不同于其他行业，这个行业每时每刻都产生着大量数据，由于安防行业对市场相对比较敏感，因此传统安防企业纷纷布局智能安防系统，在智能监控以及安保机器人等方面加大投入。安防行业产生的视频数据是非常珍贵的资源，近两年在人工智能技术的发展之下，这些数据的意义正在不断发生变化。

深度学习技术的两个重要领域包括语音识别和计算机视觉，这两个技术领域对于已经掌握海量视频和图像资源的安防行业而言，契合程度非常高，而且更容易将人工智能技术实现商业化变现。

就智能安防行业而言，深度学习可以帮助安防行业进行音频分析、图像分析和视频分析。在图像分析方面，可以进行人脸识别、车牌识别、文字识别、场景识别、身份核对，并且深度学习可以有效提升图像分析的准确度；在音频分析方面，可以进行语音识别、声纹识别等，提升安防安全等级；在视频方面，可以帮助进行活体检测、行为预测等。

就目前而言，深度学习在安防行业的应用主要集中在人体分析和车辆分析、行为分析等。其中人体分析主要包括人脸识别、人体特征抽取；车辆分析主要包括车辆识别技术、车辆特征抽取；行为分析主要包括目标跟踪监测技术、异常行为分析等。但是随着深度学习的持续发展，未来在目标识别、场景分割、物体属性分析等方面也有望取得重大突破。目前，在计算机视觉领域发力的企业包括商汤科技、格灵深瞳、神州云海等。

智能安防与传统安防技术手段相比，结合人工智能、云计算和大数据、物联网的智能安防技术，已经开始多元化快速发展，使得安防领域更加安全可靠。

1.3.3　无人驾驶

无人驾驶本身并不新鲜，最初的无人驾驶可以追溯到 1925 年，由美国陆军电子工程师通过无线电操控一辆汽车的驾驶，包括方向盘、离合器、制动器等汽车部件。虽然离完全自动化驾驶有相当大的差距，但它就是无人驾驶的雏形。后来无人驾驶技术有多次发展，但是无人驾驶不同于智能驾驶。

近些年，无人驾驶能够得到较好的发展，源于社会需求和人工智能技术的长足发展。

（1）社会需求。人们的生活已经离不开汽车，随着汽车保有量的不断增加，对环境带来的污染问题、交通事故等问题已经开始凸显，新能源汽车虽然在不断发展，但是需要更多的新技术去提升汽车给人类带来的安全性和环保性。

（2）技术发展。人工智能技术发展到当前，已经可以将一些不可能的事情变为可能。

当前，学术界将汽车智能驾驶分为五个等级：

（1）纯人工驾驶。驾驶员完全控制汽车的操作和行驶。

（2）特定功能辅助驾驶。例如针对一些特殊的防抱死制动系统、电子稳定性控制等。

（3）组合功能辅助驾驶。例如自动循环功能、车道保持辅助系统等。

（4）有限无人驾驶。能够在特定的交通环境中实现无人驾驶，例如沙漠公路等。

（5）完全无人驾驶。完全无人驾驶是无人驾驶的目标，整个过程均不需要人工参与。

从现状来看，大多数的智能驾驶水平处于组合功能辅助驾驶和有限无人驾驶的中间阶段。

目前很多公司都在深入无人驾驶，例如，百度在 2017 年百度 AI 开发者大会上开放了阿波罗计划，将无人驾驶技术推向各大汽车厂商，并预计在 2020 年能够达到有限无人驾驶；同时，前百度自动驾驶事业部总经理王劲表示，景驰无人驾驶预计在三年后（2020 年）后投放量产；2017 年 8 月，神州人生态系统“优车智脑”也正式亮相，使得共享出行也跨入人工智能时代。

在美国，除谷歌、Uber、特斯拉外，租车服务公司 Lyft 也成立了相应的无人驾驶部门，在加州投资建设制造工厂，招募数百位工程师与其他公司共同研发无人驾驶系统。美国国会初步放宽无人驾驶的限制，快速催生市场发展。

而传统的汽车厂商，也在逐步发力无人驾驶。2017 年，丰田人工智能研究院发布了首款无人驾驶汽车原型；2016 年 8 月福特宣布将在 2021 年之前实现无人驾驶汽车的量产；国内的一汽、上汽、广汽等车企也正在布局无人驾驶领域。

无人驾驶并不会立刻替换家庭的私家车，而是会从公共交通、快递用车、工业应用等领域逐步进入家庭。人工智能技术在计算汽车行驶过程中的智能识别和智能控制尤为重要，例如，道路中的车道识别、红绿灯识别、行人识别，等等。根据车的“视觉”对车辆行驶进行有效的控制。

1.3.4　电商零售

“新零售，新电商”是目前电商零售领域的热词，传统企业的线下模式与互联网生态的线上运营模式的深入融合趋势已经非常明显，在消费大升级的整体背景下，“新零售”将成为电商未来发展的重要战场。

人工智能技术已经在电商领域影响了电商企业的仓储物流管理、智能导购服务、智能客服等。传统电商模式采用用户搜索和发现的方式购买产品，这种方式已经被日益发展的云计算技术、大数据技术和人工智能技术所替代。取而代之的是针对用户行为的个性化需求产品以及带有刺激用户产生购买行为的产品。目前而言，电商会对自身平台的用户进行用户画像，这些画像包括年龄、性别、工作、消费能力、购物习惯等，为用户建立消费模型，精准推送产品，免去了用户漫无目的地搜索商品过程，使得商品的购买成功率更高。

人工智能技术不仅可以直接影响购物行为，还能影响购物体验。例如，在直播平台或者短视频、电视剧中植入智能广告技术，挖掘视频中的消费可能。对视频的物体进行分析检测对比，精准识别服装品牌、明星同款等，并为用户实现导购服务，使得商品的购买体验焕然一新。

除此之外，利用人工智能技术使得智能客服具备同理心的高情商，无疑会提升电商和零售的客服个性化服务水平。不仅如此，对于线下销售的实体店，通过摄像头捕获用户表情和行为数据也会带来不一样的体验。例如，分析实体店中哪个区域对于用户的吸引力较大，或者客户在什么情况下会选择放弃购买产品等。前几年零售业热衷于利用免费 WiFi 进行客流分析，但这种方式不仅效果差，而且存在法律风险。

未来人工智能技术还将为电商零售业带来更多机会，不断完善用户画像、优化推荐方式、优化物流管理、强化场景营销等，不仅能够综合提升电商零售的营销能力，而且更加满足用户的体验和需求。

1.3.5　智慧医疗

智慧医疗是目前医疗行业的热词，主要是通过人工智能技术打造智能导诊机器人，建立电子病历，并进行医学影像分析等，目的在于改善就医环境，减轻医生压力，提升看病就医过程中的工作效率和医疗诊断准确率。人工智能辅助诊疗系统可在医生看病、治病过程中提供推荐性建议和辅助分析，综合提升医生的诊断服务能力，在目前人口高龄化、医疗人员短缺、医疗费用较高等背景下，具有非常重要的意义。目前，在智慧医疗领域的主要应用场景包括医疗健康诊断、智能医疗设备等。

相比前几年的“互联网+”模式，现在已经正式进入了“人工智能+”时代，智慧医疗倍受关注也是理所当然，医疗是挽救生命、延长寿命和提高生存质量的重要保障，不同于其他行

业，医疗行业是每一个人都非常关注的领域。智慧医疗结合计算机视觉技术、OCR 智能识别系统等，并结合海量的医疗诊断历史数据对诊断结果进行分析，甚至有人提出“没有 AI，就没有精准医疗”，人工智能技术对于智慧医疗必会产生深远影响。

虽然目前智慧医疗整体而言，依然属于辅助医疗的范畴，但各大人工智能公司仍在不断发力智慧医疗。例如，微软亚洲研究院发起了智慧医疗项目，是以人工智能和大数据作为基础，研究领域包括基础的医学自然语言理解、基于计算机视觉与机器学习技术的数字医学影像识别，以及利用语音识别和自然语言理解技术所进行的医疗文字处理等；科大讯飞与安徽省立医院成立了人工智能辅助诊疗中心，科大讯飞的 AI 系统学习了近百万张医学影像资料、53 本专业医学教材、200 万份去标识化真实电子病历、40 万份医疗文献及病历报告，目前已正式投入应用，为 41 家县级医院提供人工智能远程辅助诊疗。

值得说明的是，人工智能技术与医疗的结合不仅仅是一次技术的结合，更是未来缓解医患关系的重要保障，让技术辅助医疗，最大程度降低误诊、漏诊等现象是建立医者与患者的信任基础。

1.3.6 金融服务

就金融行业而言，最核心的内容就是数据，而人工智能技术最需要的就是数据，因此人工智能技术可以帮助金融行业完成自动化的基础数据研究、量化交易以及智能投顾等，对于金融行业，未来信审、建模、基础数据员、分析员等，这类机械重复率高且创造力低的工种将会被人工智能技术优先淘汰，人工智能技术已经成为金融行业的重要组成部分。

目前而言，人工智能在金融领域不断取得突破，从金融产品研发到产品的推荐，以及智能数据分析、在线客服等，已经开始不断将金融服务提升到智能化、个性化阶段。从另外一个角度来看，人工智能技术也在不断影响金融安全，包括风险分析、风险控制等。智能投顾是人工智能对金融服务影响最直接的一面，在金融市场不断深入发展的过程中，金融产品的层次与交易策略以及交易工具越来越复杂，对于一般的投资者而言，难以学习和使用，而专业的投资顾问服务费用比较高、服务流程复杂且不易于实时交流，无法满足一般投资者的需求，而基于人工智能技术的智能投顾可以成为投资者的投资顾问。

当然，也可以借助人工智能技术对投资者进行定向分析，例如对其风险承受能力、收益期望、投资偏好等进行分析，综合这些分析，可以对其投资内容进行组合优化，提供投资参考，并对市场行业的动态进行个性化定制，从而成为千人千面的个性化服务。目前互联网金融中获取用户的成本非常高，而人工智能技术提供的专业化、个性化的金融产品和服务，已成为达到较高用户留存率的手段之一。

对整个金融行业而言，风险控制是互联网金融的重要管理手段。例如，对于贷款而言，传统的风险控制是审查央行的征信记录以及贷款偿还能力等，而借助大数据和人工智能技术，可以对贷款人进行身份认证、欺诈行为识别、社会关系评估等，辅助风险控制，蚂蚁金服的芝麻信用分也是基于用户画像和行为记录构建的信用体系，对于风险控制也起到了积极作用。

目前人工智能技术在金融领域还不够深入，随着技术的不断发展，人工智能技术将会更好地应用于金融领域，毕竟金融领域是唯一纯数字领域。

1.3.7　智能教育

在智能教育领域，人工智能技术主要应用场景包括智能评测、个性化辅导、儿童陪伴服务等。目前而言，已经利用人工智能技术实现了自动化作业批改、拍照搜题、英语口语测评、个性化学习方案设计等业务，这些均隐藏着人工智能技术在图像识别、OCR 识别技术、语义分析、语音识别等领域的深入发展。

从目前的现状来看，人工智能技术对教育行业的影响还远远不够，未来人工智能将会继续借助计算机视觉技术、语音识别以及自然语言处理技术等，改变传统的教育方式，至少包括以下几个方面：

（1）个性化的学习体验，因材施教；

（2）辅助分析学习内容，自动化构建知识图谱；

（3）自动化辅导、学习方案设计、在线测评以及智能答疑；

（4）模拟和虚拟化教学方式，生成情景交融的学习方式；

（5）一般教育阶段划分为幼儿早教、K12、高等教育和职业教育，辅助教育机器人在这几个教育阶段都可以做到辅助学习的作用。

教育行业不同于其他行业，可以快速接入一些新的事物进行改变。教育是国之根本，人工智能技术不会一蹴而就地改变教育行业，但是人工智能与教育结合的趋势正在持续蔓延和融合，虚拟现实（VR）技术也将结合人工智能技术对教育行业产生积极影响。人工智能不仅是对教育方式的影响，甚至会引发教育革命，未来很多工种都有面临消失的风险，只有促进教育的改革，才有可能培养出适合未来的新型劳动力，而且这样的教育方式更加注重多元化、创造力、沟通力以及学习能力等。

1.4 未来猜想

作为一线工作的人工智能研究者，我们认为人工智能技术对于传统思想也有一定的影响，人工智能的发展必然会影响人的认知变革、文化变革、思想变革。

1.4.1 人文的快速发展

人文是人类进化过程中最核心的部分，是属于人本身的特殊文化，由于人工智能技术目前仍处于发展阶段，且不会在短时间内具备人文特质，而人工智能技术促进社会的工业发展毋庸置疑，因此人文的价值在未来必定会被凸显，未来人文的发展将会成为新的突破，“重理轻文”的观念也将会打破。

在人工智能快速发展的阶段，人文社科领域的学者很容易被视为人工智能的被动参与者，但笔者认为其实恰恰相反，他们将会是人工智能发展历程上的重要参与者。人工智能并不仅仅是自然科学中快速发展的独有领域，也是人文科学与自然科学的交叉领域，人文科学的缺失会导致人工智能未来的发展处于不健康的状态。从宏观上而言，人工智能技术属于科技行业，它促使了社会发展，但是人工智能技术并不能深刻地理解科技发展对人类社会的复杂影响。

除此之外，对于普通大众而言，随着人工智能技术的发展，未来越来越多的自动化机器会替代人工劳动力，会使得普通大众有更多的时间和机会去发展人文艺术，例如美术、文学、舞蹈、音乐等，人文则会在学者和普通大众中更加流行。人文艺术领域是难以被目前的人工智能技术所替代的领域，甚至未来一段时间在人文方向的艺术、哲学、人类学等都不完全具备替代的可能。目前的人工智能技术尚处于判定分析类技术，对于感性、多领域协作还处于初级阶段，但并不意味着不可能。

1.4.2 人类也是“机器人”

深度神经网络的内部看似一个“黑盒”的结构，但并不是说结果充满随机性。结果的随机性仅仅是对认为深度神经网络是黑盒的研究者而言的，只要对计算机的结果可重复，那也只是超越那些研究者的认知而已。

作为一个长期从事深度神经网络的开发者，作者认为：随着研究的持续深入，人类越来越发现自己不过是一个“机器人”。

一般大家都说“机器人没有情感，是冰冷的机器”，但是问题在于人的情感是如何产生和表达的，机器目前只能模拟人类简单的行为，但是并不意味着机器不能模拟人的情感和思考方式。人的信息传递依靠生物神经元，正是基于生物神经元，人类研发了神经网络模型，并能够模拟人

类的部分行为，虽然并不是全部，但是随着对人体本身研究的深入，机器模拟并不是不可能。

人的行为发生，总是有一定原因的。例如，饿了的时候，人会选择吃饭；困了的时候，人会选择睡觉。人的行为事件之间充满原因和结果的关系，或者说事件之间的相互影响导致人做出了某些行为。宏观来看，不同人的自我学习过程和表现形式都非常相似。如果以单个人作为研究对象，某个人之所以能够获得某领域的诺贝尔奖，是源于他在该领域的努力，他之所以努力是因为从小就受到该领域熏陶，而他受到熏陶根源在于他的家庭长期以来的影响，这种类似的关联关系，影响了人最终的行为和结果。而这些关联关系，偶然之间充满必然，这是未来人工智能需要持续研究的问题，直至长期模拟人的行为和结果。如果这个命题成立，则人们常说的“缘分”“巧合”“机会”都是“命中注定”的，似乎一切事情都“冥冥之中，自有天意”。

基于这样的思考，人类也许是带有具备遗传特性的“机器人”，时间会逐步给我们答案。人工智能研究到最后，最可怕的不是证明人工智能多么厉害，机器崛起也并不重要，而是不断证明人多么像一个机器，其实人生都是已知的，作者觉得这才是伦理上的危机。

1.4.3　新的不平等现象

人工智能技术的发展未来会远远超过人类的想象，每一个人都能快速从人工智能技术中受益，但是并不代表着每一个人的受益是一样的。新的科学技术的出现，倘若管理不善或使用不当，则会导致甚至加剧不平等现象的风险，人工智能技术一旦脱离控制，就会导致更多无法想象的后果。

未来人类更多关注的是生活的意义和人生的价值，而不再是温饱问题。工业革命以来，人类试图用机器替代人类劳动，人类的劳动被陆陆续续替代，机器的拥有者也获得了更高的财富。人工智能技术作为解放生产力的工具是否和工业革命以来的机器一样，使得创造的财富向少数人收拢。从某种角度来看，随着人工智能技术的发展，技术的核心可能会被少数精英所控制，掌握核心则很可能掌握更多的社会财富，会导致其在社会中的实权较高，不平等现象会逐步显现。当然，如果从另外的角度来看，人工智能技术也会使得人与人之间更加的平等，例如生产更多粮食，使得每个人都有食物，人人享有温饱的机会，每个人生存的权利得到保障，但这也仅限于在不平等的基础上解决平等问题，人的需求总是在不停地转换，在解决温饱问题之后，其他的不平等问题则会出现。

除了人与人之间的不平等现象的出现，也会伴随国与国之间不平等现象的扩大，未来掌握人工智能核心技术的国家可能会更具备国际竞争力。但总体而言，技术的变革对于劳动力的影响会被各种各样的社会制度系统所调节，减少不平等现象的方法则是确保人工智能技术符合整体人类的利益，而非少部分人的利益，或者减少少数人对技术拥有的垄断。人工智能技术应当尽量不被少数人利用，人工智能技术给社会大众的应当是重塑工作的价值和意义、财富创造的

新方式，以及人生真实的含义。

1.5 本章小结

本章从基础概念出发，介绍了深度学习与机器学习、人工智能的关系，并介绍了神经网络与人工智能的发展历史。目前人工智能已经影响着各大应用领域的重塑，人工智能深入到各大领域必然会改变行业格局，传统的方式将会被淘汰。此外，人工智能的发展对人文发展、人类的认知也会产生一定的影响。

人工智能正在紧密成为人类生活的一部分，也在重塑各行各业的传统形态，影响着人类思想进步的意识，因此对待人工智能技术，应抱以一种坦然和宽容的方式，它的进步焦点不在于它能否超过人类的智慧，而在于是否能帮助人类更好的生活。

第 2 章

数学理论基础

数学是基础学科，深厚的数学基础是深度了解深度学习的重要知识储备，可以毫不夸张地说，要想设计一个独特的深度学习网络结构，缺乏数学理论的支持是几乎不可能的事情，数学深刻影响着深度学习的发展。

2.1 向量

向量，也被称为矢量，是数学、物理学和工程学中的基本概念，它指的是同时拥有大小和方向的几何物体，通常以***箭头***符号命名，以区别于其他量（例如标量）。矢量通常被标记为带有箭头的线条，线条的长度用来表示向量的大小，箭头所指的方向则是向量的方向。与向量概念相对的是标量，它只有大小没有方向。

2.1.1 相关概念

（1）反向量：大小相等，方向相反的向量。

（2）零向量：始点与终点重合的向量。

（3）等向量：大小相等且方向相同的向量。

（4）点积：点积也被称作数量积，是向量与向量乘积的结果，点积的值为标量。倘若 $\boldsymbol{A}$ 向量与 $\boldsymbol{B}$ 向量是任意向量，则它们的点积计算公式如下：

$$\boldsymbol{A} \cdot \boldsymbol{B} = |\boldsymbol{A}||\boldsymbol{B}|\cos\theta$$

通过上述公式，可以将点积理解为$\boldsymbol{A}$向量在$\boldsymbol{B}$向量上的投影大小。

其他的向量乘法还包括向量积，它也是向量与向量之间的乘积，不过它的结果依然为向量。

2.1.2 向量的线性相关性

对于n个向量$\boldsymbol{a_1}$、$\boldsymbol{a_2}$、$\boldsymbol{a_3}\ldots\boldsymbol{a_n}$，若存在$n$个不全为 0 的$p_1$、$p_2$、$p_3\ldots p_n$，使得$\sum_{i=0}^{n} p_i\boldsymbol{a_l}=0$，则意味着这$n$个向量$\boldsymbol{a_1}$、$\boldsymbol{a_2}$、$\boldsymbol{a_3}\ldots\boldsymbol{a_n}$是线性相关的。同理，若是不存在这样的$n$个不全为 0 的$p_1$、$p_2$、$p_3\ldots p_n$，当且仅当$p_i$=0 时才成立，则可以称这$n$个向量$\boldsymbol{a_1}$、$\boldsymbol{a_2}$、$\boldsymbol{a_3}\ldots\boldsymbol{a_n}$线性无关。

线性相关和线性无关的理论是机器学习乃至神经网络的重要基础。

2.1.3 向量的外积

外积一般是指两个向量的张量积，其结果为矩阵。向量的外积可以视为矩阵的克罗内克积的特殊形式。

对于给定$m\times 1$的列向量$\boldsymbol{u}$以及$1\times n$的行向量$\boldsymbol{v}$，两者的外积可以用$\boldsymbol{u}\otimes\boldsymbol{v}$表示，且两者的外积结果为$m\times n$的矩阵。若设定$m$的值为 4，则$\boldsymbol{u}$可以表达为如下形式：

$$\boldsymbol{u}=\begin{bmatrix}u_1\\u_2\\u_3\\u_4\end{bmatrix}$$

若设定n的值为 3，则$\boldsymbol{v}$的表达形式可以为$\boldsymbol{v}=[v_1 \quad v_2 \quad v_3]$，则$\boldsymbol{u}\otimes\boldsymbol{v}$的结果如下：

$$\boldsymbol{u}\otimes\boldsymbol{v}=\begin{bmatrix}u_1\\u_2\\u_3\\u_4\end{bmatrix}\otimes[v_1 \quad v_2 \quad v_3]=\begin{bmatrix}v_1u_1 & v_2u_1 & v_3u_1\\v_1u_2 & v_2u_2 & v_3u_2\\v_1u_3 & v_2u_3 & v_3u_3\\v_1u_4 & v_2u_4 & v_3u_4\end{bmatrix}$$

2.1.4 向量夹角与余弦相似性

余弦相似性，顾名思义就是通过余弦的方式计算相似度。余弦是两个向量的夹角，因此可以将需要进行相似性比较的内容视为余弦的两个向量，若是两个向量大小方向完全重合，即夹角为 0°，余弦值为 1；同理，若两个向量完全相反，即夹角为 180°，则余弦值为-1，向量a和向量b的夹角示例如图 2-1 所示。因此，余弦相似性是基于向量的算法，并利用两个向量在方向和大小上的不同，将此种不同转换为余弦值，并将余弦值用作衡量信息差异的标准。

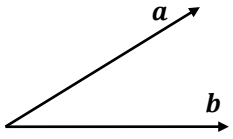


图 2-1 两个向量的夹角示例

此处的余弦值大小即可表达出两个向量之间的相似度，这也是余弦相似性算法能够进行相

似度计算的基本原理，余弦相似性常常用于文本内容的比较。

计算向量的$\boldsymbol{a}$（x_1，y_1）和向量$\boldsymbol{b}$（x_2，y_2）的余弦值，可采用如下公式：

$$\mathrm{COS}(\theta)=\frac{\boldsymbol{a}\times\boldsymbol{b}}{||\boldsymbol{a}||\times||\boldsymbol{b}||}=\frac{(x_1,y_1)\times(x_2,y_2)}{\sqrt{x_1^2+y_1^2}\times\sqrt{x_2^2+y_2^2}}$$

上述公式是针对二维向量的计算，在实际工作中，很可能是 n 维向量的计算，计算公式如下所示：

$$\mathrm{COS}(\theta)=\frac{\boldsymbol{a}\times\boldsymbol{b}}{||\boldsymbol{a}||\times||\boldsymbol{b}||}=\frac{\sum_{i=1}^{n}(x_i\times y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2}\times\sqrt{\sum_{i=1}^{n}(y_i)^2}}$$

2.1.5　实例：基于向量夹角的文本相似性分析

向量夹角可以衡量相关向量的相似，基于此可以通过余弦相似性对一部分问题求解。采用余弦相似性进行相似度分析，可以参考表 2-1 中的内容，通过余弦相似性计算文本之间的相似程度。

表 2-1　利用余弦相似性求句子相似性

句子编号	句子内容
句子 A	深度神经网络是一种深度学习方法
句子 B	神经网络是一种学习方法

由于文本的相似性在于文本中词语之间的相似性，因此可以将词语在文本中出现的频率初步作为词语在文本中的权值。

（1）分词处理。对句子 A 和句子 B 进行分词处理，句子 A 分词后结果为“深度 神经网络是 一种 深度 学习方法”；句子 B 分词后结果为“神经网络 是 一种 学习方法”。在实际工程中，可以利用 TF-IDF 算法或者 TextRank 算法等提取关键词，包括停用词处理等，而不仅限于对分词的结果。

（2）获得向量集。根据上述两个句子的分词结果，将所有词语进行合并，相同的词语则仅记一次，因此获得向量集“深度 神经网络 是 一种 深度 学习方法”。

（3）计算词频。针对句子 A 和句子 B 得到的公共向量集“神经网络是一种学习方法”。对句子 A 和句子 B 计算词频，存在则记录词频，不存在则记为 0。因此对于句子 A 可计算词频“深度[2]神经网络[1]是[1]一种[1]学习方法[1]”；对于句子 B 可计算词频“神经网络[1]是[1]一种[1]学习方法[1]”。

（4）形成特征向量。根据特征集计算的词频信息，将词频转换为特征向量，特征向量以特

征集为基础，词频为特征集对应词语的权值，因此针对句子 A 特征向量为“2 1 1 1 1”，句子 B 的特征向量为“0 1 1 1 1”。

（5）相似度计算。现在已经获得特征向量{2，1，1，1，1}和{0，1，1，1，1}，因此问题可以转换为对向量求夹角，根据余弦夹角公式，计算过程如下所示：

$$\mathrm{COS}(\theta)=\frac{2\times0+1\times1+1\times1+1\times1+1\times1}{\sqrt{2^2+1^2+1^2+1^2+1^2}\times\sqrt{0^2+1^2+1^2+1^2+1^2}}$$

通过上述过程，最终计算出余弦值为 0.7，因此，可以认为句子“深度神经网络是一种深度学习方法”与“神经网络是一种学习方法”的文本相似度为 0.7。

2.2 矩阵

矩阵是高等代数中的常用工具，在应用数学学科如统计分析中非常普遍。

2.2.1 矩阵乘法

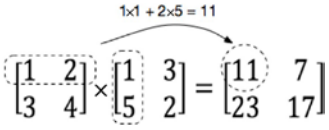
最简单的矩阵乘法是一个常数乘以矩阵，例如常数 3 乘以矩阵$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$的计算结果如下：

$$3\times\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}=\begin{bmatrix}1\times3 & 2\times3\\3\times3 & 4\times3\end{bmatrix}=\begin{bmatrix}3 & 6\\9 & 12\end{bmatrix}$$

但是在更多深度学习的计算过程中，是矩阵与矩阵相乘，例如矩阵$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$与矩阵$\begin{bmatrix}1 & 3\\5 & 2\end{bmatrix}$相乘的计算结果如下：

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\times\begin{bmatrix}1 & 3\\5 & 2\end{bmatrix}=\begin{bmatrix}11 & 7\\23 & 17\end{bmatrix}$$

直观上不好理解计算方式，实质上是矩阵的行乘以被乘矩阵的列，如下所示：



从这个公式可以清晰地看到，被乘结果的矩阵第m行第n列的元素值等于第一个矩阵的第m行乘以第二个矩阵的n列的元素乘积求和。

在线性代数中，矩阵乘法可以表示线性方程之间的关系，例如方程组：

$$\begin{cases}x+2y=3\\5x+3y=8\end{cases}$$

可以用矩阵乘积表示为如下：

$$\begin{bmatrix}1 & 2\\ 5 & 2\end{bmatrix}\times\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}3\\ 7\end{bmatrix}$$

因此，任意的方程组都可以用矩阵表示。

2.2.2　克罗内克积

克罗内克积表示两个任意矩阵之间的积运算，用符号⊗表示。倘若矩阵 $\boldsymbol{A}$ 为 $p*q$ 的一个矩阵，$\boldsymbol{B}$ 为任意矩阵，则矩阵 $\boldsymbol{A}$ 和矩阵 $\boldsymbol{B}$ 的克罗内克积如下列公式所示：

$$\boldsymbol{A}\otimes\boldsymbol{B}=\begin{bmatrix}\mathrm{a}_{11}\boldsymbol{B} & \cdots & \mathrm{a}_{1\mathrm{q}}\boldsymbol{B}\\ \vdots & \ddots & \vdots\\ \mathrm{a}_{\mathrm{p}1}\boldsymbol{B} & \cdots & \mathrm{a}_{\mathrm{pq}}\boldsymbol{B}\end{bmatrix}$$

例如若 $\boldsymbol{A}$ 矩阵为$\begin{bmatrix}1 & 3\\ 2 & 0\end{bmatrix}$，$\boldsymbol{B}$ 矩阵为$\begin{bmatrix}0 & 1\\ 2 & 2\end{bmatrix}$，则两者的克罗内克积计算过程如下所示：

$$\begin{bmatrix}1 & 3\\ 2 & 0\end{bmatrix}\otimes\begin{bmatrix}0 & 1\\ 2 & 2\end{bmatrix}=\begin{bmatrix}1\times0 & 1\times1 & 3\times0 & 3\times1\\ 1\times2 & 1\times2 & 3\times2 & 3\times2\\ 2\times0 & 2\times1 & 0\times0 & 0\times1\\ 2\times2 & 2\times2 & 0\times2 & 0\times2\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 3\\ 2 & 2 & 6 & 6\\ 0 & 2 & 0 & 0\\ 4 & 4 & 0 & 0\end{bmatrix}$$

克罗内克积与数值乘法的结合律很相似，例如 $\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$ 为三个任意矩阵，则有如下规律：

（1）$\boldsymbol{A}\otimes(\boldsymbol{B}+\boldsymbol{C})=\boldsymbol{A}\otimes\boldsymbol{B}+\boldsymbol{A}\otimes\boldsymbol{C}$

（2）$(\boldsymbol{A}+\boldsymbol{B})\otimes\boldsymbol{C}=\boldsymbol{A}\otimes\boldsymbol{C}+\boldsymbol{B}\otimes\boldsymbol{C}$

（3）$(k\boldsymbol{A})\otimes\boldsymbol{B}=\boldsymbol{A}\otimes(k\boldsymbol{B})=k(\boldsymbol{A}\otimes\boldsymbol{B})$

（4）$(\boldsymbol{A}\otimes\boldsymbol{B})\otimes\boldsymbol{C}=\boldsymbol{A}\otimes(\boldsymbol{B}\otimes\boldsymbol{C})$

但是克罗内克积不满足数值乘法的交换律，在一般情况下$A\otimes B\neq B\otimes A$。

除此之外，克罗内克积还有部分混合乘积的性质，例如矩阵 $\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$、$\boldsymbol{D}$，且矩阵乘积 $\boldsymbol{AC}$ 和 $\boldsymbol{BD}$ 均存在，则$(\boldsymbol{A}\otimes\boldsymbol{B})(\boldsymbol{C}\otimes\boldsymbol{D})=\boldsymbol{AC}\otimes\boldsymbol{BD}$。

对于转置运算克罗内克积也满足，公式为：$(\boldsymbol{A}\otimes\boldsymbol{B})^{\mathrm{T}}=\boldsymbol{A}^{\mathrm{T}}\otimes\boldsymbol{B}^{\mathrm{T}}$。

在深度学习中，涉及比较多的矩阵计算，例如，在卷积神经网络中图像的输入是像素，从数学角度就是矩阵，涉及矩阵的各类转换或计算。从向量计算扩展到矩阵的运算是特征获取和计算的另外一种方式。

2.3 导数

2.3.1 概述

导数是微积分中最基本的理论，一个函数在某一个位置的导数表示了该函数在该位置下的变化率。导数实质上是通过极限的方式对函数进行局部的线性逼近。通常用$f'(x)$表示$f(x)$的导数，$f(x)$在区间内的连续性是函数可导的必要条件。导数的特性如下。

（1）如果某函数的导数在某个取值区间内的值均大于 0，则函数在当前区间属于单调递增；同理，如果均小于 0，则函数在当前区间属于单调递减。

（2）如果函数在当前某值下的导数为 0，左区间是单调递增，右区间是单调递减，则当前函数的当前值为极大值点；同理，若左区间是单调递减，右区间是单调递增，则当前函数的当前值为极小值点。

导数的计算方式相对比较简单，一些常见函数的导数如表 2-2 所示。

表 2-2 一些常见函数的导数

函数 $f(x)$	导数 $f'(x)$
常数 c	0
$\sin(x)$	$\cos(x)$
$\cos(x)$	$\sin(x)$
x^n	nx^{n-1}
a^x	$a\text{x}\ln a(a>0)$
e^x	e^x
$\log_a x$	$\frac{1}{x\ln a}$（$a>0$，且 a 不等于 1）
$\ln x$	$\frac{1}{x}$

2.3.2 一般运算法则

导数的一般运算规则如下。

（1）两个函数的和或差的导数均等于两个函数分别求导之后的和或差，公式如下：

$$[f(x) \pm g(x)]' = f'(x) \pm g'(x)$$

（2）两个函数乘积的导数，等于 f 函数的导数乘以 g 函数与 g 函数的导数乘以 f 函数之和，公式如下：

$$[f(x)\times g(x)]' = f'(x)\times g(x) + g'(x)\times f(x)$$

（3）两个函数商的导数，等于 f 函数的导数乘以 g 函数减去 g 函数的导数乘以 f 函数之差，然后再除以 g 函数的平方（g 函数不等于 0），公式如下：

$$[f(x)/g(x)]' = \frac{f'(x)\times g(x) - g'(x)\times f(x)}{g(x)^2}$$

2.3.3　链式求导法则

对于一些复杂的函数求导，可以采用链式法则求导法则。链式法则（Chain Rule）是求复合函数导数的一个法则。若f与g是符合函数$(f\circ g)(x)$的导数，则$(f\circ g)'(x)$计算公式为：

$$(f\circ g)'(x) = f'(g(x))g'(x)$$

例如，求解函数$f(x) = (x^2+6)^3$的导数，设$g(x) = x^2+6$，$h(x) = g(x)^3$，即$f(x) = h\big(g(x)\big)$，则计算$f'(x)$导数的过程如下：

$$f'^{(x)} = h'\big(g(x)\big)g'(x) = 3(g(x))^2\times 2x = 3(x^2+6)^2\times 2x = 6x(x^2+6)^2$$

2.4　数值计算

2.4.1　误差

在有监督的模型训练过程中会涉及模型的输出值与期望输出值的差异，这种差异用误差来表示，并且根据误差不断调整训练过程的参数，最终的目标是整体样本中模型的实际输出结果与期望数据结果的误差尽可能小。

（1）均方差（Mean Squared Error，MSE）。均方差是统计学中较为常见的误差计算方式，它是误差的平方的期望，是衡量平均误差的一种较为简便的方法，可以有效地评价数据的变化情况。

（2）标准误差（Standard Error），也称均方根误差（Root Mean Squared Error），是衡量样本分布的离散度，是样本均值与总体均值的差异。对于一个总体样本的多次抽样，每次抽样的大小均相同，每次抽样的数据都有一个均值，这些均值的标准差则被称作标准误差。

标准误差越小，则样本均值和总体均值越相近，反之越大。标准误差常常用于预测样本数据的有效性或准确性，当标准误差越小时，则样本均值和总体均值差距越小，当差距小于某阈值时，样本数据则可代表总体数据情况。

标准误差是针对样本统计量而言，是对某个样本统计量的标准差，是描述对应的样本统计

量抽样分布的离散程度以及衡量对应样本统计量抽样误差大小的尺度。

（3）标准差（Standard Deviation）则是描述样本整体的离散程度。标准差的值越大则说明数据分布越广，集中程度较差，均值的代表性则越差；反之，若标准差的值越小，则均值的代表性越强。

2.4.2 距离

数值距离的计算是机器学习算法中对分析结果非常重要的衡量标准，是单次抽样的结果。在机器学习算法的应用过程中，常常涉及一些数值计算，总体而言主要集中在两方面：一方面是距离计算；另一方面是概率计算。距离和概率是机器学习算法中最为核心的数值，是表达信息异同相似的数值体现。常见的距离计算方式包括欧氏距离、马氏距离、曼哈顿距离、切比雪夫距离、海明距离，等等。

（1）欧几里得度量（Euclidean Metric，也称欧氏距离）。欧氏距离是一个常用的距离定义，指在 m 维空间中两个点之间的真实距离，或者向量的自然长度（即该点到原点的距离）。在二维和三维空间中，欧氏距离就是两点之间的实际距离。

例如，对于二维平面上的两点$a(x_1, y_1)$与$b(x_2, y_2)$之间的欧氏距离公式为

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

同理，对于三维平面上两点$a(x_1, y_1, z_1)$与$b(x_2, y_2, z_2)$间的欧氏距离公式为

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

欧氏距离是距离算法中最常用的方式，日常生活中的大部分场景都可以通过欧氏距离进行计算。

虽然欧氏距离比较常用，但是缺点也比较明显，它是将数据的特征进行独立的计算，且差别是等同的，这个缺陷导致部分现实问题无法通过欧氏距离计算获得。例如，在心理学上，需要对人的性格进行分析，但是人的性格对于不同的个体是不同的，也与人的其他属性有着很大的关系，因此不能进行简单的欧氏距离计算。

（2）马氏距离（Mahalanobis Distance）。马氏距离是由印度统计学家马哈拉诺比斯提出的一种表示数值协方差距离的概念。这种协方差距离体现的是数据样本分布的距离。与欧氏距离不同的是，它考虑到各种特性之间的联系，并且是尺度无关的，即独立于测量尺度。马氏距离可用于计算两个未知样本信息集合的相似度分析。

（3）曼哈顿距离（Manhattan Distance）。曼哈顿距离是由十九世纪的赫尔曼·闵可夫斯基

所创的一个词汇，是一种使用在几何度量空间的几何学用语，用以标明两个点在标准坐标系上的绝对轴距总和。它表示的不是两点间的直线距离，而是实际上从 A 点到达 B 点的距离，例如城市街区中，很多时候我们不可能通过直线的方式到达目的地，原因也很简单，我们无法穿越建筑，而是通过街道到达的。

对于二维平面两点$a(x_1, y_2)$与$b(x_2, y_2)$间的曼哈顿距离为：

$$d = |x_1 - x_2| + |y_1 - y_2|$$

（4）切比雪夫距离（Chebyshev Distance）。切比雪夫距离是向量空间中的一种度量，两个点之间的距离定义为其各坐标数值差的最大值。以(x_1, y_2)和(x_2, y_2)两点为例，其切比雪夫距离为$\max(|x_1 - x_2|, |y_1 - y_2|)$。切比雪夫距离得名自俄罗斯数学家切比雪夫。对于两个n维的变量$\boldsymbol{a}(x_1, x_2, x_3 \ldots, x_n)$与变量$\boldsymbol{b}(y_1, y_2, y_3 \ldots, y_n)$之间的切比雪夫距离如下：

$$d = \lim_{k \to \infty} \left(\sum_{i=1}^{n} |x_i - y_i|^k \right)^{1/k}$$

（5）闵氏距离（Minkowski Distance）。闵氏距离又叫作闵可夫斯基距离。对于两个n维的变量$\boldsymbol{a}(x_1, x_2, x_3 \ldots, x_n)$与变量$\boldsymbol{b}(y_1, y_2, y_3 \ldots, y_n)$之间的闵可夫斯基距离的定义如下所示，其中$p$是一个变参：

$$d = \sqrt[p]{\sum_{k=1}^{n} |x_k - y_k|^p}$$

当上述公式中$p = 1$时，实质就是曼哈顿距离；当$p = 2$时，则是欧氏空间中的一种测度，被看作是欧氏距离的一种推广，欧氏距离是闵可夫斯基距离中p等于 2 的一种特殊情况；而当p趋近于无穷大时，则可以视为切比雪夫距离。

（6）海明距离（Hamming Distance）。在信息论中，两个等长字符串之间的海明距离是两个字符串对应位置的不同字符的个数。换句话说，它就是将一个字符串变换成另外一个字符串时所需要替换的字符个数。

海明重量是字符串相对于同样长度的零字符串的海明距离，也就是说，它是字符串中非零元素的个数：对于二进制字符串来说，就是 1 的个数，所以 11101 的海明重量是 4。例如，“1011111”与“1011001”之间的海明距离是 2；“2243896”与“2533796”之间的海明距离是 3；“bejing”与“peking”之间的海明距离是 2。

对于固定的长度n，海明距离是该长度字符向量空间上的度量，很显然它满足非负、唯一

及对称性，并且可以很容易地通过完全归纳法证明它满足三角不等式。两个字符a与b之间的海明距离也可以看作特定运算$a-b$的海明重量。

2.4.3 数值归一化

归一化（Normalization）的目的是让每一个对象在不同分类上的权值对应到区间[0，1]或[–1，1]。归一化是一种数据预处理方法，就是把需要处理的数据经过特定算法处理后，限制值区间在一定范围之内，这样做不仅是为了后面数据处理的方便，也是为了保证程序运行时加快收敛。一般常见的归一化方法如下所示。

（1）10 为底的对数函数。对数函数的公式为$f(x)=\lg(x)$。

（2）min-max 归一化。min-max 归一化方法是对原始数据的线性改变，使得其结果映射在[0，1]，min 表示数据集里的最小值，max 表示数据集里的最大值。对于任意一个x，它的归一化公式如下：

$$f(x)=\frac{x-\min}{\max-\min}$$

（3）z-score 归一化。z-score 归一化的方法借助了数据集中平均值和标准差进行计算，处理之后的数据集符合正态分布的特点，均值为 0，标准差为 1。对于任意的x，它的归一化公式如下，其中μ是样本均值，σ是样本的标准差：

$$f(x)=\frac{x-\mu}{\sigma}$$

除上述介绍的数值归一化方法外，其他常见的归一化方法还包括$f(x)=\text{atan}(x)\times 2/\text{PI}$、$\log_{10}^{(x+1)/(\max(x)+1)}$等。

2.5 概率分布

算法常常涉及数据分布情况，常见的分布方式包括二项分布（Binomial Distribution）、超几何分布（Hypergeometric Distribution）、泊松分布（Poisson Distribution）以及正态分布（Normal Distribution）。

2.5.1 二项分布

二项分布表示n个独立的是与非试验中成功次数的离散概率分布情况，每一次的成功或者失败试验都被称为伯努利试验。例如，已知某生产线产品的合格率为 90%，则重复抽取产品线中的产品，连续k次且每次仅抽取一件，则k次抽取得到的合格产品个数x的分布即为二项分布。

值得说明的是，当$n = 1$时即为伯努利分布。

一般情况下，如果随机变量x服从参数n和p的二项分布，则n次试验中刚好得到k次成功的概率质量函数如下：

$$f(k,n,p) = \binom{n}{k} p^k (1-p)^{n-k}$$

例如，进行抛硬币的实验，已知抛得正面和反面的概率均为 0.5，总共抛 10 次，可能结果有 0 次正面到最多 10 次正面，则恰好有n次为正面的二项分布的示例如图 2-2 所示。

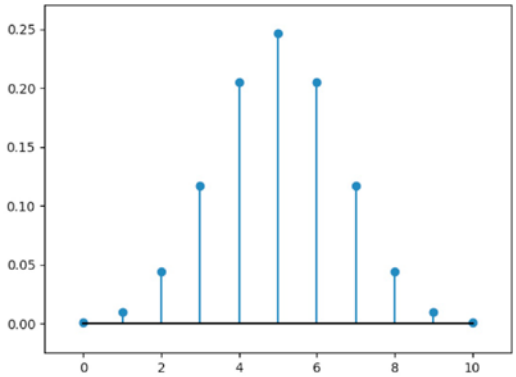


图 2-2　抛硬币正面向上的二项分布示例

2.5.2　超几何分布

超几何分布也是统计学中一种离散概率分布，和二项分布类似，不同点在于样本的量。当n=1 时，超几何分布即为伯努利分布；而当 n 的数量不可估量或趋近于无穷时，超几何分布即为二项分布。超几何分布是在有限的样本中的试验。例如，某公司有 100 名工程师，其中 30 名女性、70 名男性，现需要外派 10 名工程师外出学习，被选到女性工程师的人数 x 是一个随机变量，而 x 的分布即为超几何分布。

2.5.3　泊松分布

泊松分布适合描述在单位时间内随机事件发生的次数的概率分布。例如，搜索引擎在某时间内的请求次数、某时间内火车站进站人数等。泊松分布与二项分布有一定的关联性，二项分布可以视为泊松分布在离线时间上的对应。泊松分布的概率质量函数如下，其中参数λ是单位时间内随机事件的平均发生率。

$$p(k,\lambda) = \frac{\mathrm{e}^{-\lambda}\lambda^k}{k!}$$

例如，已知某公交车每天往返的平均发生率为 3 次，则可以绘制出每天往返 0 到 10 次的概率，如图 2-3 所示。

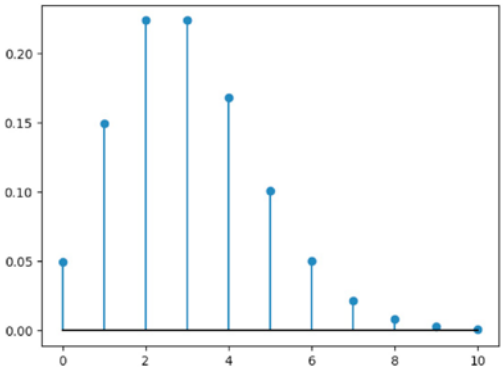


图 2-3　公交车往返次数的泊松分布示例

2.5.4　指数分布

指数分布是一种连续概率分布，指数分布可以用于分析独立的随机事件发生的时间间隔。例如，一般人喝水的时间间隔、城市下雨的时间间隔，等等。它与泊松分布也存在一定关联，泊松过程是一种随机过程，而在泊松分布过程中，第 N 次随机事件与第 N+1 随机事件的时间间隔是符合指数分布的。一个指数分布的概率密度函数如下，其中$\lambda > 0$是分布的一个参数，常被称作率参数，可以理解为每单位时间内发生该时间的次数：

$$f(x,\lambda) = \begin{cases} \lambda e^{-\lambda x}, x \geqslant 0 \\ 0, x < 0 \end{cases}$$

一个$\lambda = 0.5$，x取值（0，20）的指数分布大致如图 2-4 所示。

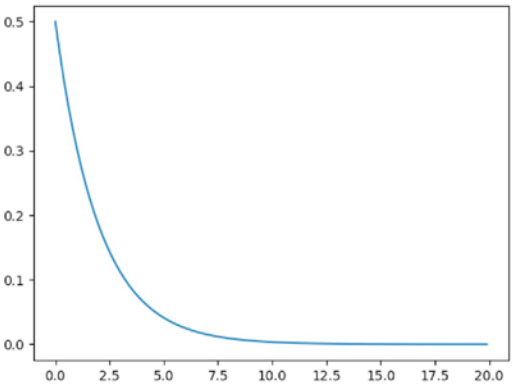


图 2-4　指数分布的示例

2.5.5　正态分布

正态分布也被称作高斯分布，是概率论中非常重要的分布，简单的理解即为当某个数量的指标受到大量随机因素的影响，然而每一个因素对结果产生的影响又非常小时，则该数据指标可以视为服从正态分布。正态分布中通过概率密度函数对随机变量进行分析，很多算法都依赖于正态分布进行估算。

正态分布作为一种连续的函数分布，应用非常广泛，其计算公式如下，其中μ为平均值，σ为标准差：

$$p(x,\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

例如，一个简单的正态分布效果如图 2-5 所示。

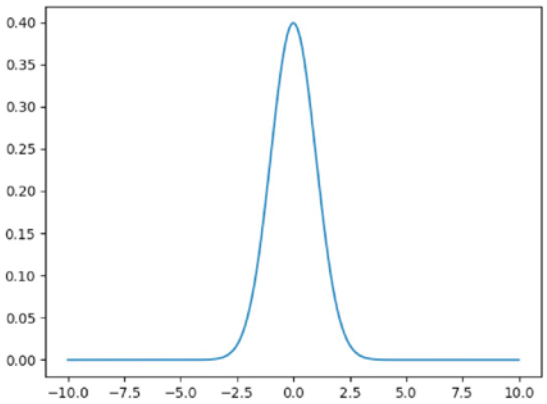


图 2-5　一个简单的正态分布示例

2.6　参数估计

2.6.1　概率

概率论中有很多内容也是算法中常常涉及的问题。尤其在概率相关的概念中，常常会见到如下概率：条件概率、先验概率、后验概率、联合概率，它们的概念如下。

1. 条件概率

从统计的角度去分析数据的分布情况，对两个随机变量 X 和 Y，其联合分布是同时对于 X 和 Y 的概率分布。在一系列的事件发生中，如果 A 事件与 B 事件都有发生，若 A 事件的发生

与 *B* 事件的发生存在某种关系，则这种关系可表述为当 *B* 事件发生的情况下，*A* 事件可能发生的概率，即表达为 *P*(*A*|*B*)，表示在 *B* 事件发生的前提下，发生 *A* 事件的条件概率，同时此刻的 *B* 事件的值则被称作 *A* 事件的后验概率。

2. 先验概率

先验概率又被称作边缘概率，在一系列的事件发生中，如果 *A* 事件与 *B* 事件都有发生，则 *P*(*A*)即可被称作 *A* 事件的先验概率，先验概率是一种相对独立的概率，*A* 事件的先验概率则不依赖于 *B* 事件的发生。

3. 联合概率

已知两个相关的随机变量 *X* 和 *Y*，随机变量 *Y* 在条件{*X*=*x*}下的条件概率分布是指当已知 *X* 的取值为某个特定值 *x* 时，*Y* 的概率分布。联合概率是指两个事件同时发生的概率。在一系列事件发生中，如果 *A* 事件与 *B* 事件都有发生，则 *P*(*AB*)表示 *A*、*B* 两个事件同时发生的概率。

2.6.2 贝叶斯估计

在利用朴素贝叶斯（Naive Beyesian）解决问题之前，首先需要了解贝叶斯定理（Bayes' Theorem），因为朴素贝叶斯是基于贝叶斯定理为基础的概率分类模型。贝叶斯定理是概率论中的一个定理，它与随机变量的条件概率以及边缘概率分布有关。在关于概率的某些解说中，贝叶斯定理能够告知我们如何利用新证据修改已有的看法。这个名称来自于托马斯·贝叶斯。

通常，事件 *A* 在事件 *B*（发生）的条件下的概率，与事件 *B* 在事件 *A* 的条件下的概率是不一样的；然而，这两者间有着确定的关系，贝叶斯定理就是对这种关系的表示。贝叶斯公式定义在事件 *B* 出现的前提下，事件 *A* 出现的概率等于事件 *A* 出现的前提下事件 *B* 发生的概率乘以时间 *A* 出现的概率再除以时间 *B* 出现的概率。通过联系事件 *A* 与事件 *B*，计算从一个事件产生另一事件的概率，即从结果上溯源，因此贝叶斯定理公式如下所示：

$$P(A|B)=\frac{P(B|A)\times P(A)}{P(B)}$$

在理解上述贝叶斯定理的基础上，则可以较好地理解基于朴素贝叶斯的分类模型。信息分类是信息处理中最基本的模块，每一段信息无论长或短，它都是由若干特征组成的，因此可以将所有特征视为一个向量集$\boldsymbol{W}=(\boldsymbol{w}_1,\boldsymbol{w}_2,\boldsymbol{w}_3,\dots,\boldsymbol{w}_n)$，其中$\boldsymbol{w}_i$即表示其中第$i$个特征。而信息的分类也可以视为一个分类标记的集合$C=(\boldsymbol{c}_1,\boldsymbol{c}_2,\boldsymbol{c}_3,\dots,\boldsymbol{c}_m)$。在进行特征学习之前，特征$\boldsymbol{w}_j$与分类标记$\boldsymbol{c}_j$的关系不是确定值，因此需要提前计算$P(\boldsymbol{C}|\boldsymbol{W})$，也就是当特征$\boldsymbol{w}_i$出现的情况下，信息属于分类标记 *C* 的概率，概率可根据贝叶斯计算，公式如下：

$$P(\boldsymbol{C}|\boldsymbol{W}) = \frac{P(\boldsymbol{W}|\boldsymbol{C})\times P(\boldsymbol{C})}{P(\boldsymbol{W})}$$

因此可以从信息分类的角度理解贝叶斯公式，即表示为：在特征$\boldsymbol{w_i}$出现的情况下是否是特征类别$\boldsymbol{c_j}$取决于在特征分类标记$\boldsymbol{c_j}$情况下特征$\boldsymbol{w_i}$出现的概率，以及$\boldsymbol{w}_i$在所有特征中出现的概率。$P(\boldsymbol{W})$的意义在于如果这个特征在所有信息中出现，那么用特征$\boldsymbol{w_i}$去判定是否属于分类标识$\boldsymbol{c_j}$的概率越低，越不具备代表性。

2.6.3　最大似然估计

最大似然估计（Maximum Likelihood Estimate，MLE）是统计学中常用的概率计算工具，它在试图解决这样一个问题：给一组样本数组以及一个固定模型，但是模型的参数是未知的，通过确定这些未知参数使得模型在已知样本中的概率值最大，即确定在某种参数情况下已知事件的发生概率最大。

一般而言，先验概率是在了解原因的情况下求结果；后验概率是在了解结果的情况下求原因；最大似然估计是知道结果的基础上求最有可能的原因。例如，已知机场流量控制会导致某航班延误，这里的机场流量控制是原因，而某个航班延误则是结果。先验概率计算的是机场流量控制的情况下，会导致某个航班延误的概率；后验概率计算的是某航班已经发生延误，则可能属于机场流量控制的概率；最大似然估计是某航班已经发生延误，则计算最大可能导致该航班延误的原因，因为有可能还有目的地天气、飞机晚到等原因。

对于最大似然估计，可以通过一个例子进行深入了解。例如，在某个黑色的盒子中，里面有橙色和白色的乒乓球，橙色乒乓球与白色乒乓球的比例未知，通过不断地从盒子中拿出一个乒乓球，并记录其颜色，然后将球再放回盒子中，重复上述过程，最终通过记录的乒乓球颜色比例以估计盒子中的橙色乒乓球与白色乒乓球的比例。目的是通过若干次的实验得到某个参数值，使得在该参数值的情况下，样本发生的概率值最大。

给定一个概率分布D，假定它的概率密度函数或者概率质量函数为f_D，以及一个分布参数θ，如果从这个分布中抽取出n个样本$x_1, x_2, \dots, x_n$，然后通过f_D，计算采用的概率：

$$p(x_1, x_2, \dots, x_n) = f_D(x_1, x_2, \dots, x_n|\theta)$$

其中概率密度函数针对的是连续分布，而概率质量函数针对的是离散分布。对于未知的θ值，可以通过抽取的n个样本进行估算θ值。最大似然估计在于估算θ最可能的值，换个角度即当θ为某个值时，会使得采用样本$x_1, x_2, \dots, x_n$的概率最大化。

下面通过一个实例以了解最大似然估计法的核心思想：“已知模型，参数估计”。例如，有三个不透明的盒子 A、B、C，里面均有若干橙色乒乓球和白色乒乓球，已知 A、B、C 盒子中

随机取出橙色乒乓球的概率分别是 2/3，3/5，2/7。现在在 A、B、C 三个盒子中随机取了一个盒子，通过对这个盒子进行取乒乓球的颜色实验，每次取完均放回盒子，一共进行了 100 次取球的实验，其中 40 次是橙色乒乓球，60 次是白色乒乓球，则这个盒子最可能属于 A、B、C 中的哪一个？

$$P(A)=(\frac{2}{3})^{40}\times\left(1-\frac{2}{3}\right)^{60}$$

$$P(B)=(\frac{3}{5})^{40}\times\left(1-\frac{3}{5}\right)^{60}$$

$$P(C)=(\frac{2}{7})^{40}\times\left(1-\frac{2}{7}\right)^{60}$$

通过上述计算可得 $P(C)$的概率最大，因此随机取的盒子最可能是盒子 C。最大似然估计方法的原理非常简单，很容易在实际工作中使用到，但是当样本数量较少时，则可能导致计算结果的误差较大。例如，在从盒子取乒乓球的例子中，如果只进行了 10 次实验，那么计算结果很可能与实际产生偏差。最大似然估计的主要优点包括：

（1）随着样本数的不断增加，它的收敛性将会明显变好，因为实验样本越多，越接近真实值。

（2）原理比较清晰，实现过程也相对比较简单。

2.6.4 最大后验估计

在贝叶斯统计中，最大后验估计（Maximum A Posterior，MAP）是通过经验数据对难以分析的点进行估计，它与最大似然估计的方法有一定的类似，但是最大后验估计引入了被估计参数的先验估计，从这个角度可以将最大后验估计视为最大似然估计的正则化。虽然最大后验估计利用了贝叶斯估计的思想，但是它不能全面地表示贝叶斯估计，因此它不属于贝叶斯估计，这是因为最大后验估计采用的是点估计，而贝叶斯估计则是借助分布情况进行结论推测，贝叶斯估计更倾向于后验估计的均值或者中位数，类似于一种可信区间，而不再是一个点。

对于最大后验估计，可以通过以下示例进行了解：假设有五个盒子，每个盒子中均有一定的硬币，硬币有 5 角的，也有 1 元的，各自数量如表 2-3 所示。

表 2-3 盒子中不同硬币类型的数量

盒子编号	5 角硬币数量	1 元硬币数量
1	100 枚	0 枚
2	75 枚	25 枚
3	50 枚	50 枚
4	25 枚	75 枚
5	0 枚	100 枚

基于表 2-3，如果连续从一个盒子拿到两枚 1 元的硬币，求最有可能的盒子编号。采用最大似然估计的方式比较容易得到答案，假定从盒子中能够拿出 1 元一枚硬币的概率为p，则同时拿出两枚的概率是p^2。因此，分别计算p为：0%、25%、50%、75%、100%；两枚的概率是p^2，依次计算p^2为：0%、6.25%、25%、56.25%、100%，因此很可能是从第 5 个盒子中取出的。

上述问题中，每个盒子被拿到的概率是均等的，因此在上述问题的基础之上，新增扩展条件：能够拿到盒子 1、2、3、4、5 的概率不相等，且每个盒子被拿到的概率如表 2-4 所示。

表 2-4　盒子被取到的概率分布

盒子编号	取到概率（g）
1	0.1
2	0.2
3	0.4
4	0.2
5	0.1

在此条件下，连续从一个盒子中拿到两枚 1 元的硬币，求最有可能的盒子。此问题则从最大似然估计的问题转换为最大后验概率的问题，根据最大后验概率公式：

$$\mathrm{MAP} = p^2 \times g$$

而已知p为：0%、25%、50%、75%、100%，g分别为 0.1、0.2、0.4、0.2、0.1，依次进行推算可以得出来自第 4 个盒子的概率最大。

2.7　回归分析

2.7.1　线性回归

线性回归（Linear Regression）是统计学中对若干的样本数据的自变量与因变量之间线性关系的一种回归模型，利用数理统计分析的方法，确定两个或者多个变量之间依赖的定量关系的一种统计分析。

在线性回归中，数据采用线性函数的方式进行数据建模，对于模型中的未知参数也采用数据进行估计。对于一个多变量的线性回归模型可以表示为如下公式：

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip} + \varepsilon_i,\ \ i = 1,2,\ldots,n$$

其中 y 是 x 的线性函数，ε_i是误差项。Y_i 的条件均值在参数β里是线性的。有些函数模型

看起来并不一定是线性回归，但是通过代数转换可以转换为线性回归模型。对于一个简单的线性回归，可以表示为：

$$Y_i = \alpha + \beta x_i$$

单变量的线性回归是目前最容易理解的线性回归，其回归公式如下所示：

$$Y = \alpha + \beta x + \varepsilon$$

线性回归是回归分析中使用最为广泛的模型，在结果预测以及函数关系中应用较为频繁。

最小二乘法是线性回归的一种典型方法，也被称作最小平方法。最小二乘法的基本原则是：最优拟合的直线应该是各个点到直线的距离之和尽可能小，即平方和最小。在这个过程中，使用得最多的是欧几里得距离，因为它的计算简单，是一种较好的相似度计算方式。平方损失函数的标准形式如下：

$$L\big(Y, f(X)\big) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

公式中，$y - f(x)$表示残差，$L\big(Y, f(X)\big)$表示的是残差的平方和，通过计算上述的损失值，应当尽可能减小损失，即最小化残差的平方和。

一条多元一次的直线方程，在二维坐标中即二元一次方程。例如，在二维坐标中，有非常多的点分散在其中，试图绘制一条直线，使得这些分散的点到直线的距离最小。这里的距离最小并非点到直线的垂直距离最短，而是点到直线的 y 轴距离最短，即该点到该 y 轴平行线与直线交点的距离最短，如图 2-6 所示的双向箭头。

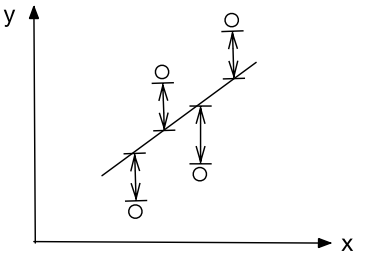


图 2-6　最小二乘法中点到直线的距离

最小二乘法的核心思想是通过最小化误差的平方和，试图找到最可能的函数方程。例如，在二维坐标系中存在五个数据点（10,20）、（11,23）、（12,25）、（13,27）、（14,26），希望找出一条距该五个点距离最短的直线，根据二元一次方程$y = ax + b$，将五个点分别代入该二元方程得到：

$$20 = 10a + b$$

$$23 = 11a + b$$

$$25 = 12a + b$$

$$27 = 13a + b$$

$$26 = 14a + b$$

由于最小二乘法是尽可能使得等号两边的方差值最小：

$$S(a,b) = [20 - (10a + b)]^2 + [23 - (11a + b)]^2 + [25 - (12a + b)]^2 + [27 - (13a + b)]^2 + [26 - (14a + b)]^2$$

因此求最小值即可通过对 $S(a,b)$求偏导数获得，并使得一阶导数的值为 0：

$$\frac{\partial S}{\partial a} = 1460a + 120b - 2936$$

$$\frac{\partial S}{\partial b} = 12a + 10b - 242$$

即得到关于求解未知变量 a、b 的二元一次方程：

$$\begin{cases} 1460a + 120b = 2936 \\ 12a + 10b = 242 \end{cases}$$

通过计算上述二元一次方程即得到a=0.0243，b=24.1708。因此，在上述五个点中，通过最小二乘法得到直线方程：$y = 24.1708 + 0.0243x$，是使得五个点到该直线距离最小的直线。

最小二乘法虽然看似是一个直线方程的问题，但在实际应用中应用非常广泛，因为它得到的方程可以视为一个函数模型，该函数模型可以给后续的工作带来极大的便利。例如，某种疾病是在两种条件下发生的，但是需要当这两种条件满足一定关系时才会促发疾病，因此医生就可以通过患病样本获得患病情况下的两种条件值，并标记到一个二维坐标中，通过最小二乘法，可以将患病的两种条件通过函数表达出来，从而当有另外一个新疑似患者就医时，就可以根据二元一次方程确定是否可能患有该疾病。

上述过程均是通过线性问题的求解方式进行阐述的，但是更多的时候，需要解决的问题不是一个线性问题，它需要通过多项式拟合的方式进行处理，但是原理和求解方式均一致。虽然最小二乘法易于实现，在各行各业中都被广泛使用，但是它的计算量也比较大，当样本数据不断增加后，计算量会明显增加，在阶数更高时计算量则更为复杂。为了解决更多问题，基于最小二乘法衍生出了移动最小二乘法、加权最小二乘法以及偏最小二乘法等。

除最小二乘法外，梯度下降也是线性回归中的方法之一。线性回归目前已经被广泛应用在

计算机科学领域，除此之外还在金融、经济学等其他领域也被应用，是数值统计分析的较好工具。

2.7.2 逻辑回归

逻辑回归（Logistics Regression）是一种分类方法，逻辑回归分为二元逻辑回归和多元逻辑回归。对于二元逻辑回归即给定一个输入，输出 True 或 False 来确定它是否属于某个类别，并给出属于这个类别的概率。逻辑回归简单、高效，应用非常广泛，在分类模型中占据非常重要的地位。对于二元逻辑回归，一般可以帮助解决类似以下问题：

（1）预测用户是否购买某些商品。

（2）在用户画像中，确定用户的性别。

（3）预测用户是否点击某个链接。

（4）判断用户评论属于正面评论还是负面评论。

逻辑回归的数学描述如下。

输入为x，输出为y，中间有一个临时变量为t，w和b为模型参数，公式如下所示：

$$t = wx + b$$

$h(t)$属于此类别的概率大于 0.5 时认为y=1，一般采用 Sigmoid 函数作为转换函数，Sigmoid 函数的公式如下所示：

$$h(t) = \frac{1}{1 + \mathrm{e}^{-t}}$$

Sigmoid 的函数曲线如图 2-7 所示，Sigmoid 函数的取值区间在（0,1）。

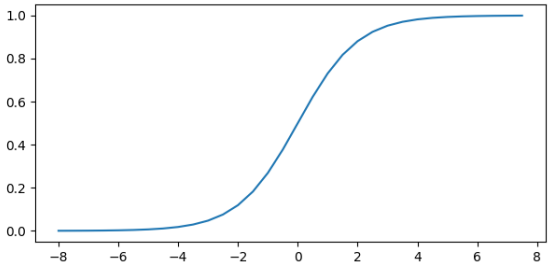


图 2−7 Sigmoid 函数曲线

根据t和$h(t)$的公式，可以将 Sigmoid 函数的公式化简为逻辑回归的公式，如下：

$$h(x)=\frac{1}{1+\mathbf{e}^{-wx}}$$

对于t函数中的参数b，可以理解为b乘以一个值为 1 的w_0，因此可以将b化简到x的系数w中。

逻辑回归与神经网络也存在一定的关系。假设对于一个四维的输入样本，即$x=\{x_1,x_2,x_3,x_4\}$，则在不考虑偏置项b的前提下，模型的训练参数w也与输入样本的维度一致，即$w=\{w_1,w_2,w_3,w_4\}$，则逻辑回归可以用图 2-8 表示。

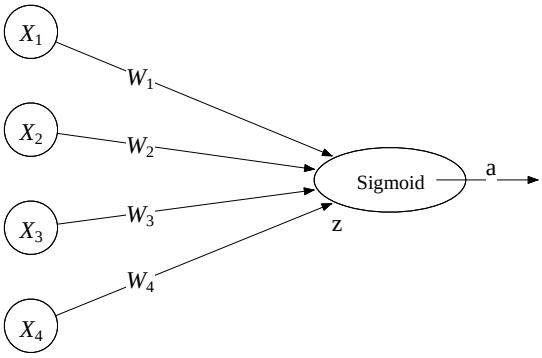


图 2-8　逻辑回归的图形化表示

左边为原始的四维特征输入向量。连接边缘表示模型参数w用于表示输入向量对应权值，即输入特征向量在每个维度上的权值。

权值与输入特征向量的内积结果则是Sigmoid函数的输入下项，Sigmoid的作用和神经网络的激活函数相似，Sigmoid的输出项则是一种后验概率。

上述是逻辑回归的简单表示，实际上可以将多个逻辑回归的输出作为另一个逻辑回归分类器的输入，并且让逻辑回归分类器负责输出分类任务的类别。图 2-9 采用两个逻辑回归分类器的输出作为另一个逻辑回归分类器的输入。

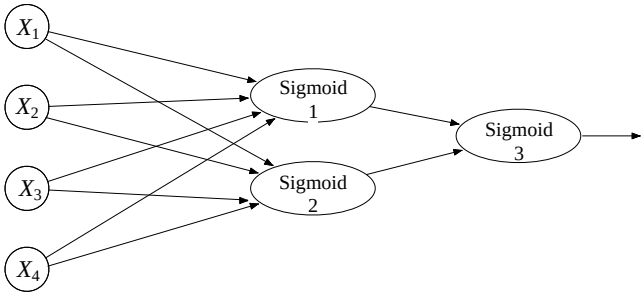


图 2-9　两个逻辑回归的输出作为另外一个逻辑回归分类器的输入

带有Sigmoid 1的逻辑回归分类器和带有Sigmoid 2的逻辑回归分类器的输出分别是a_1和a_2，同时a_1和a_2是Sigmoid 3逻辑回归分类器的输入。最终模型的输出是二元分类任务中类别的后验概率。

将不同的逻辑回归进行组合，有助于解决非线性划分问题。例如，对于下面这个分类任务，将实心圆样本和空心圆样本进行划分，简单的单个逻辑回归分类器只能划分为如图 2-10 所示。

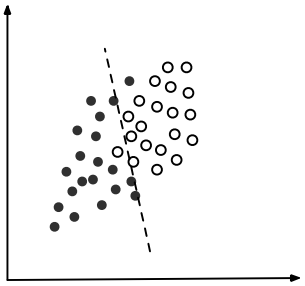


图 2-10　单个逻辑回归的分类划分示意

而对于图 2-9 中的三个逻辑回归组合，其中带有Sigmoid 1的逻辑回归可能进行的划分如图 2-11 所示。

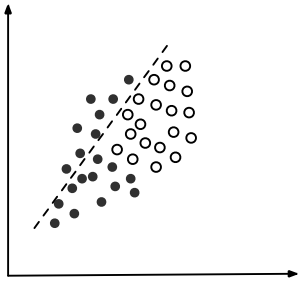


图 2-11　带有 Sigmoid 1 的逻辑回归划分示意

而对于带有 Sigmoid 2 的逻辑回归可能进行的划分如图 2-12 所示。

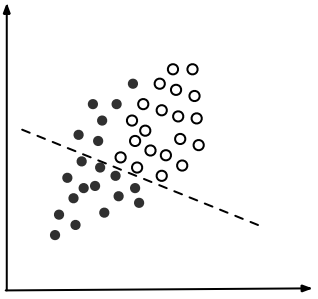


图 2-12　带有 Sigmoid 2 的逻辑回归划分示意

如果将带有Sigmoid 1的逻辑回归划分和带有Sigmoid 2的逻辑回归划分进行结合，则可能获得的划分如图 2-13 所示，有效地解决了非线性分类问题。

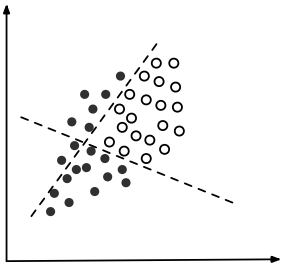


图 2-13　合并两个逻辑回归划分的最终结果示意

原本线性的逻辑回归分类器，通过组合后成为一个复杂的非线性分类器，这也是神经网络解决的问题。

逻辑回归的数学模型和求解方式都比较简单，逻辑回归可以处理各类非线性问题，作为一个强大的分类器，当遇到一些低层次的特征问题时，可以优先考虑用逻辑回归来解决。

2.8　判定问题

2.8.1　P 问题

P（Polynomial，多项式）问题是一种具有多项式算法的判定性问题，表示可以在多项式时间内解决的所有决策问题的集合。倘若一个判定性问题的解决复杂度为一个实例规模 n 的多项式函数，则此类可以在多项式时间内解决的判定性问题属于 P 问题。简单而言，即给出问题的一个实例，可以在多项式时间内决定答案是或否。

多项式时间（Polynomial Time）属于计算复杂度理论中的常见概念，指的是一个问题的计算时间 $m(n)$不大于问题大小 n 的多项式倍数。

P 问题比较直观，我们平常求解的问题大多数都是 P 问题，例如数组排序、最短路径求解等。

2.8.2　NP 问题

NP（Non-Deterministic Polynomial，非确定多项式）问题是指可以在多项式时间内被解决的非决定性问题。对于 NP 问题，可能没有一个确定已知的方法解决问题，但是可以通过候选答案尝试验证的方式在多项式时间内得到该问题的解，从这个角度来看，P 问题是 NP 问题的一个子集。同时，NP 问题不一定都是难解的问题。

2.8.3 NP-Complete 问题

现在已知 P 问题是可以在多项式时间内解决的问题，NP 问题是可以在多项式时间内验证一个解的问题，而 NP-Complete（Non-Deterministic Polynomial Complete，NPC）问题是在 NP 问题的基础之上发展而来的，NPC 问题是 NP 判定问题中的一个子集。

NPC 问题引入了规约的概念（Reducibility），规约可以简单理解为，一个问题可以转换为另外一个问题，或者问题的求解可以借鉴另外一个问题的求解。例如，若一元二次方程是具备求解方式的，则一元一次方程的求解方式也应该是具备的。通过规约可以了解到，一个问题 A 规约为另外一个问题 B，则问题 B 的时间复杂度不低于问题 A 的时间复杂度，也就是问题 B 相对而言更难。求解一元二次方程的难度比求解一元一次方程的难度高，因此可以借鉴求解一元二次方程的方式进而求解一元一次方程。此外，值得注意的是，规约具备传递的特性。

满足以下两个特征的问题则属于 NPC 问题：

（1）该问题属于 NP 问题。

（2）一个已知的 NPC 问题能够规约到该问题。

所有的 NP 问题都能规约到 NPC 问题，关键在于一个 NPC 问题能否找到利用某个多项式算法求解，这样似乎所有的 NP 问题都能够借助该方法求解，如果这样的命题成立，则 NP 问题就成为了 P 问题。但实际上，NPC 问题目前还没有有效的多项式算法，基本只能通过指数级或者阶乘级的算法复杂度进行搜索。

2.8.4 NP-Hard 问题

如果存在这样一个问题，使得所有的 NP 问题都可以规约到此问题，则此问题被称作 NP-Hard 问题。

目前学术界对于 P 问题是否等于 NP 问题还存在争议，这种争议简单而言就是求解问题解的难度和验证一个解是否满足该问题的难度是否相同。

倘若 P 问题等于 NP 问题，则 P 问题、NP 问题、NP-Complete 问题、NP-Hard 问题的关系如图 2-14 所示。

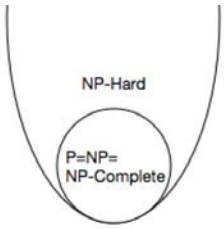


图 2-14　P=NP 情况下四者关系

倘若 P 问题不等于 NP 问题，则 P 问题、NP 问题、NP-Complete 问题、NP-Hard 问题的关系如图 2-15 所示。

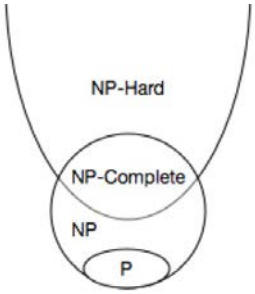


图 2-15　P!=NP 情况下四者关系

就目前而言，可以认为 P 问题是 NP 问题的子集，但是还缺乏理论验证。

2.9　本章小结

本章从向量、矩阵、导数、概率分布到参数估计、逻辑回归，再到 NP 问题进行介绍，它们都是深度学习必会用到的数学基础。数值计算是一切机器学习的基础，无论是误差计算还是距离计算；概率与分布可以适当进行辅助数据分析，贝叶斯估计、最大似然估计、最大后验估计三者各有优势，分布适用于不同的场景；回归分析则是分析解决一些实际问题，它们的底层思想和神经网络有一定的相似之处；判定问题中的 P 问题、NP 问题、NP-Complete 问题、NP-Hard 问题则可以辅助对问题进行本质分析，便于寻找问题解决方法。

第 3 章

机器学习概要

机器学习算法是基于数据基础之上，从数据中挖掘出数据规律或模型，然后利用规律或模型对未知数据进行分析或预测的算法。相对于排序算法、查找算法等，它的计算过程更多依赖于数学模型、统计和概率相关知识。除此之外，机器学习更多的是利用数据或者历史经验进行自我改善的算法。常见的机器学习算法包括决策树、神经网络、朴素贝叶斯、支持向量机等，机器学习算法可以理解为是对一些可以让计算机进行自动化“学习”算法的统称。

3.1 机器学习的类型

3.1.1 有监督学习

有监督分类是根据已有的训练集提供的样本(x_i, y_i)，通过不断计算从样本中学习选择特征参数，对分类器建立判别函数以对被识别的样本进行分类。首先它需要一定量级的训练数据，通过对训练数据进行特征提取形成符合特征的分类模型，最后将分类模型形成分类器并实现对数据的分类。

有监督的分类方式可以有效利用先验数据，对后验数据进行校验，但是缺点也比较明显，训练数据是人为收集，具有一定的主观性，并且人为收集数据也会花费一定的人力成本。另外，分类器分类的结果只能是训练数据中的分类类型，不会产生新的类型。

根据有监督学习的输出类型，可以分为回归（Regression）和分类（Classification）两种。

（1）回归问题。如果监督学习中，输出的y是一个连续值，且$f(x)$的输出也是连续的值，则此类问题可以划分为回归问题，以距离计算为主，常见的有线性回归和逻辑回归等。

（2）分类问题。如果y是离散的类别标记或者数值，则可以视为分类问题，以概率计算为主，常见的分类模型有朴素贝叶斯、支持向量机等。

对于有监督学习的机器学习，给定训练样本（x_i, y_i），其中i大于 1 且小于某个常数，x_i表

示输入参数，y_i表示期望的输出参数，机器学习是通过训练样本（x_i, y_i），让系统自动寻找出一个模型函数$f(x)$，使得x_i通过模型函数$f(x)$能尽可能输出得到对应的y_i。$f(x)$则有效建立了x_i与y_i的内在关系，一个简单的有监督机器学习系统如图 3-1 所示。

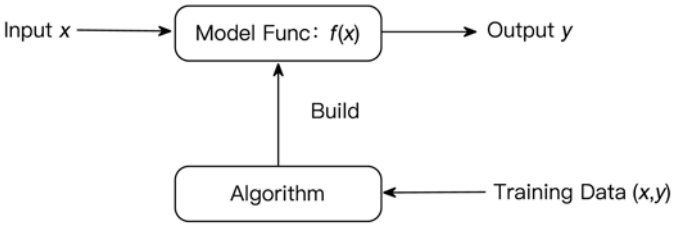


图 3-1　一个简单的有监督机器学习系统示例

3.1.2　无监督学习

无监督学习是模型本身不进行先验知识学习，不会对模型进行参数训练，而是使用被预测的样本数据直接进行预测的过程，此类预测过程只是对不同类型的数据进行了预测，预测后的结果具有不确定性。

无监督学习的方式对预测结果的选择性较大，预测结果不局限于某特定的预测类型。由于是无监督学习，所以人为干预较少，结果具备一定的客观性，但是一般无监督学习的方式计算过程较为复杂，需要大量的分析之后，才有可能获得较好的预测结果。典型的无监督学习是聚类（Clustering）。

除无监督学习外，还有半监督学习（Semi-Supervised Learning），它是建立在已有模型的基础上，通过从少量训练样本中获得先验数据的机器学习方式，结合了有监督学习和无监督学习的各自优势。

考虑到监督学习需要训练集，而训练集往往是人工参与标注，非监督学习的优势在于它不需要人工标记的数据集，因此出现了类似于弱监督学习和半监督学习的新机制，但是它们都是基于有监督学习和无监督学习，利用无监督学习从大规模的数据样本中获得有效数据，从而降低监督学习的数据标记成本，这种方式在当下的机器学习中常常被使用。

3.1.3　强化学习

强化学习主要强调的是基于当前环境下的动作行为控制，以取得最优效果，它是由一系列累计的动作行为得到的最优效果。强化学习不同于监督学习的地方在于它不需要显式地输入一些样本数据，属于一种在线的学习方式。有监督学习、无监督学习和强化学习的比较如表 3-1 所示。

表 3-1 有监督学习、无监督学习、强化学习三种方式对比

	监督学习	非监督学习	强化学习
输入	已标记的数据集	无标记的数据	决策过程
反馈	直接反馈	无反馈	奖励
用途	分类、预测等问题	发现隐藏结构，例如聚类	动作行为控制

随着深度学习的发展，传统的机器学习方式已经开始演进为有监督学习的深度网络、无监督的生成网络、混合深度网络。这里的混合是指混合系统，是将深度学习与其他机器学习组合的方式。例如，将深度学习的输出概率应用于基于隐马尔科夫模型的语音识别系统中。

3.2 机器学习中常见的函数

3.2.1 激活函数

激活函数是计算神经网络中非常重要的一环，激活函数增加了神经网络网络模型的非线性特征，倘若神经网络中不存在激活函数，那么无论神经网络的深度有多少层，最终均是若干次的矩阵相乘，若输入输出依然存在线性关系，则对于机器学习就失去了意义。

（1）线性函数（Liner Function）：

$$f(x) = a * x + b$$

其中，a和b均为常数。

（2）阈值函数（Threshold Function）：

$$f(x) = \begin{cases} 1, & x \geqslant c \\ 0, & x < c \end{cases}$$

其中，c为常数。

（3）斜面函数（Ramp Function）：

$$f(x) = \begin{cases} a, & x > c \\ k * x, & |x| \leqslant c \\ -a, & x < -c \end{cases}$$

其中，a和c为常数。

（4）S 形函数（Log-Sigmoid Function）：

$$f(x) = \frac{1}{1 + \mathrm{e}^{-\alpha x}}$$

根据上述公式，可知$f(x)$所属区间(0,1)，导函数为：$f'(x)=\frac{\alpha e^{-\alpha x}}{(1+e^{-\alpha x})^2}=\alpha f(x)[1-f(x)]$。

（5）双曲正切 S 形函数（Tan-Sigmoid Function）：

$$f(x)=\frac{2}{1+e^{-\alpha x}}-1$$

根据上述公式，可知$f(x)$所属区间(-1,1)，导函数为：$f'(x)=\frac{2\alpha e^{-\alpha x}}{(1+e^{-\alpha x})^2}=\frac{\alpha[1-f(x)^2]}{2}$

双曲正切 S 形函数与 S 形函数都是连续、单调递增的函数，差异在于取值范围，一个是(-1,1)，另外一个是(0,1)。此类可导的激活函数，非常适用于 BP 神经网络中。一般而言，BP 神经网络中的隐藏层采用的是 S 型函数，输出层采用的则是线性函数。如果输出层也采用 S 型函数，则输出值的范围取决于 S 型函数。可以利用 S 型函数或者其导数计算 BP 神经网络中某个神经元的目标值和误差值等。

（6）双曲正切函数：

$$\tanh(x)=\frac{e^{x}-e^{-x}}{e^{-x}+e^{x}}$$

（7）Softplus 函数：

$$f(x)=\log(1+e^{x})$$

Softplus 函数的导数是 Logistic 函数。

（8）Logistic 函数。

用来将某个实数映射到(0,1)，公式如下：

$$f(x)=\frac{1}{1+e^{-x}}$$

（9）ReLU 函数。

ReLU（Rectified Linear Units）函数也被称作修正线性单元，是神经网络中常见的激活函数，俗称斜坡函数，如图 3-2 所示。

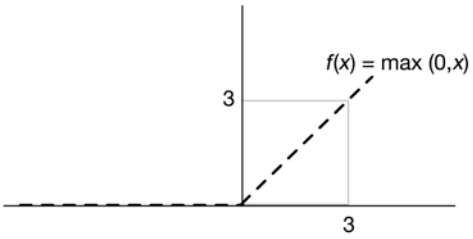


图 3-2　ReLU 函数图

同时 ReLU 函数也存在三种变异：Noisy ReLU、Leaky ReLU、Parametric ReLU。

在 ReLU 函数的基础之上加入高斯噪声可以得到 Noisy ReLU，公式如下所示：

$$f(x) = \max(0, x + N(0, \sigma(x)))$$

对于神经元的输入x，高斯噪声使得修正单元加上了一定程度的正态分布的不确定性。

对于带泄漏的 ReLU 函数（Leaky ReLU），在输入的$x < 0$时，保持一个很小的梯度λ，这样可以避免参数一直处于非激活状态。公式如下所示：

$$f(x) = \begin{cases} x, & x > 0 \\ \lambda x, & x \leqslant 0, \lambda \in (0,1) \end{cases}$$

$$= \max(0, x) + \lambda \min(0, x)$$

Leaky ReLU 主要是为接近“ReLU 死亡”问题，有些时候其表现效果比较理想，但是并不是特别稳定。λ的取值一般是在 0~1 间，例如 0.01。

对于带参数的 ReLU 函数（Parametric ReLU），公式如下所示：

$$f(x) = \begin{cases} x, & x > 0 \\ a_i x, & x \leqslant 0 \end{cases}$$

上述公式中a_i可以理解为当x小于等于 0 时的斜率，对于a_i，若a_i=0 则可以视为标准的 ReLU 函数；当a_i为一个较小的常数时，可以视为带泄漏的 ReLU 函数。

在输入值为正数时，结果保持与传统的 ReLU 函数一致；当输入值不为正数时，通过设定反向传播算法的学习变量λ，使得 ReLU 函数成为一种带参数的修正线性单元。

除了上述的 Noisy ReLU 和 Leaky ReLU 两种变异外，还存在其他的一些变异，例如 PreLU、随机 ReLU 等，最近学术圈提出了一种 SELU 激活函数，也属于 ReLU 函数的变异。它们在不同的数据集上会对模型的训练速度或准确率有所改进，在具体应用时，可根据实际情况选择不同的 ReLU 函数。

ReLU 函数给神经网络的应用带来了非常大的帮助。相比传统的激活函数，例如逻辑函数、双曲线函数等，ReLU 函数具有以下几大优势。

（1）计算量小。tanh 等激活函数，计算量比较大，在进行反向传播求误差的梯度时，求导过程涉及除法操作，计算量也比较大，但是采用 ReLU 函数，只需一个简单的判断，计算量可以大大减少，避免了指数运算。

（2）更容易学习优化。在深度学习网络中，很容易出现梯度小的情况。例如，采用 Sigmoid 激活函数进行反向传播计算，当接近饱和区时，变化趋势非常小，相应导数趋近于 0，

此时很容易造成信息丢失。而 ReLU 函数拥有分段线性的特征，会使得其前后的传递求导都是分段线性的，这是传统的 Sigmoid 等激活函数不能解决的问题。

（3）缓解过拟合问题。ReLU 函数会使得部分神经元的输出为 0，使得网络结构变得稀疏，减少了参数的相互依存关系，可以有效减少过拟合问题的发生。

当然，对于偏置值这类特殊的值，当采用 ReLU 函数时，可以在神经元开始的时候，设定到非零区域内，初始化为小的正数。

对于常用的 ReLU 激活函数，它也存在缺陷，由于小于 0 的值会被设定为 0，因此这种强制的稀疏处理会使得模型的有效信息容量减少，也就意味着部分特征可能无法有效获取。

线性函数、阈值函数、斜面函数均属于线性函数，其余为非线性函数。对于神经网络而言，每个神经元的激活函数都可以不同，但是也可以采用同一组激活函数，大多数情况下，神经元采用的是相同的激活函数。

3.2.2　损失函数

在样本训练过程中，给定一个训练集(x,y)，其中x是样本数据的训练内容，y是样本数据对应的期望内容。通过机器学习之后，得到的预测模型为$f(x)$，我们训练的目的是尽可能使得$f(x)=y$。若$f(x)$不等于 y，则产生了预测错误，因此需要一个函数来定义错误带来的损失，即损失函数。损失函数是机器学习训练过程中，进行权值调整的方式。

（1）0-1 损失函数（0-1 loss function）。

最简单的损失函数是 0-1 误差函数，公式如下所示：

$$y=\begin{cases}1 & x<0\\ 0 & x\geqslant 0\end{cases}$$

对于 0-1 误差函数而言，并不依赖于具体输入值的大小，而取决于输入值的正负，因此 0-1 误差函数只有“对”与“错”的概念，虽然足够简单，但是在具体复杂的应用中，则存在很多不足。

（2）Log 损失函数。

Log 损失函数是 0-1 误差函数的一种代理函数，公式如下所示：

$$y=\log(1+\exp(-x))$$

在分类器模型 Logistic 回归中，Log 损失函数有典型的应用。

（3）绝对损失函数：

$$L\big(y,f(x)\big)=|f(x)-y|$$

上述是几种简单的损失函数，其他的还包括平方损失函数、交叉熵损失函数等。对于损失函数而言，损失函数的值越小，表示计算结果与期望结果相差越小，此时模型也逐步趋近于较好。损失函数是对当前的输入输出进行的一次预测估算，也是对当前预测结果好坏的一个评估；而风险函数是针对当前众多输入情况平均意义下的模型优劣程度的判定。

与损失函数相似的还有能量函数，能量函数最初在热力学中定义，以描述系统的能量值。能量函数是整个系统状态的度量。 系统的有序或概率分布越多，系统的能量就越小。相反，系统更无序或趋于均匀分布，则系统的能量越大。能量函数的最小值对应于系统最稳定的状态。能量函数是反映系统稳定性的参考指标，在这个意义上与损失函数相似。

3.2.3 核函数

核函数是随支持向量机的发展演变而来的，由于支持向量机的广泛应用，核函数得到了较大的关注。核函数可以在许多应用中使用，核函数构成了从线性到非线性的关联关系，可以用点积表示。核函数隐藏了从低维空间映射到高维空间的关系，使得在低维空间中线性不可分的数据到达高维空间之后，可以线性可分。常见的核函数如下。

（1）线性核（Linear Kernel）函数。线性核函数主要应用于线性可分的场景，基于线性核的方式训练参数少，迭代速度快，对于线性分类等场景效果较好，因此对于一般线性问题采用线性核函数比较合适，其公式如下：

$$k(x,y)=x^Ty+c$$

（2）多项式核函数（Polynomial Kernel）。多项式核函数可以将低维输入空间映射到高维特征。相对于线性核函数，多项式核函数由于训练参数过多且维度较高，导致训练周期相对较长，计算复杂度较高，其公式如下：

$$k(x,y)=(ax^Ty+c)^d$$

（3）径向基核函数（Radial Basis Function）。同多项式核函数一样，它可以将一个样本映射到高维空间中，但是相对于多项式核函数，其参数较少。径向基核函数在小样本和大样本中都有较好的性能体现，应用非常广泛，其公式如下：

$$k(x,y)=\exp(-\gamma||x-y||^2)$$

同时径向基核函数也被称作高斯核（Gaussian Kernel）函数，可以视为高斯核函数的特殊情况，高斯核函数公式如下：

$$k(x,y)=\exp(-\frac{||x-y||^2}{2\sigma^2})$$

除此之外，指数核函数（Exponential Kernel）也是基于高斯核函数的变种，它的差异在于距离的调整。通过距离的调整可以减少参数依赖性，但是应用面相对受限，其公式如下：

$$k(x,y)=\exp(-\frac{||x-y||}{2\sigma^2})$$

在径向基核函数的基础之上，还产生了类似拉普拉斯核函数（Laplacian Kernel）、ANOVA 核函数（ANOVA Kernel）等核函数类型，它们之间的差异在于适应的场景不同。

（4）Sigmoid 核（Sigmoid Kernel）。Sigmoid 核函数源自于神经网络，目前在神经网络中应用非常多，常常被用作神经元的激活函数，其公式如下：

$$k(x,y)=\tanh(ax^{\mathrm{T}}+c)$$

除上述介绍的核函数外，还包括二次有理核函数（Rational Quadratic Kernel）、多元二次核函数（Multiquadric Kernel）、逆多元二次核函数（Inverse Multiquadric Kernel）、三角核函数（Triangular Kernel）等，它们大多是常见核函数的变异，以提升性能或适应特定场景。

3.3　机器学习中的重要参数

3.3.1　学习速率

在机器学习过程中，有部分参数被称作超参数，超参数是在学习训练之前设置的参数值，不同于其他参数是通过训练得到的。一般而言，机器学习的调优过程中会涉及对超参数进行调整优化，通过不断优化给出一组有效的超参数，使得模型在性能和效果上达到一个较好的结果，超参数可以简单理解为参数的参数。

常见的超参数包括梯度下降方法的步长、常规系数、神经网络的动量、RBF 内核的方差、神经网络的隐藏层层数、K-Means 聚类的数量 K 等。超参数和准确率通常是非凸的关系，因此超参数的优化是机器学习的重要课题，常用的方法是网格搜索或高斯过程。值得说明的是，超参数是神经网络训练过程中非常重要的值，神经网络中比较经典的五个超参数分别是：学习速率、权值初始化、网络层数、单层神经元数量、正则惩罚项。

学习速率（Learning Rate）是最常见的超参数之一，用于控制每次更新时调整的权值大小或权值修正的幅度，一般用η表示。如果设置学习速率太小，则会使得收敛的速度太慢；然而若是设置学习速率太大，则会导致波动较大。

学习速率或分步率是函数逐步搜索空间的速率。学习速率的典型值在 0.001 到 0.1 之间。较小的步骤意味着更长的训练时间，但可以得到更精确的结果。

3.3.2 动量系数

动量系数是确定优化算法收敛于最优点速度的另外一个因素。可以通过动量系数加速模型训练过程，但是更快的速度会降低模型的准确性。动量是在 0 和 1 之间的一个变量，被用作矩阵变化率的导数的一个因素，它影响权值随时间的变化率。

动量系数用于防止系统收敛到局部最优解中。 高动量系数也有助于提高系统的收敛速度。然而，将动量系数设置得太高可能会导致超过最小值的风险，这可能导致系统变得不稳定。过低的动量系数不能可靠地避免局部最小值，还可能减缓系统的训练速度。

在训练过程中更新权值的一个动量项，即在权值不断更改的情况下，动量可以保证权值的更改向指定的方向移动，取值在 0~1 之间。

3.3.3 偏置项

偏置项可以帮助函数进行较好的左右平移，当 b>0 时，函数向左移动；当 b<0 时，函数向右移动。例如，对于图 3-2，$y = x$可以对数据点进行很好的分割。

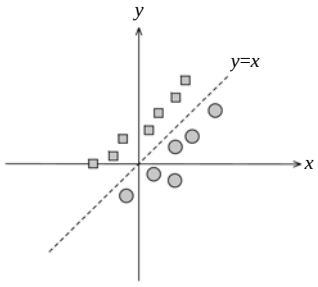


图 3-2 $y = x$对数据点的划分示例

但是如果如图 3-3 所示，则$y = x$不具备划分能力。

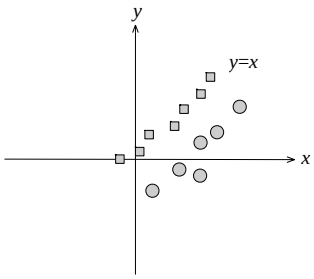


图 3-3　$y = x$不能对数据点进行划分的示例

此时需要将$y = x$向右移动才能做到有效划分，因此偏置项b的作用凸显，添加之后得到$y = x + b$，函数可以很好地在水平移动中划分数据，灵活性加强，如图 3-4 所示。

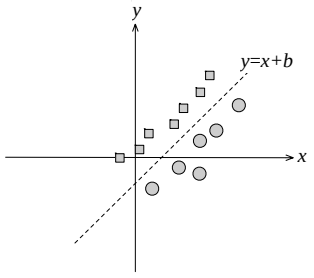


图 3-4　$y = x + b$对数据点进行划分的示例

3.4　拟合问题

3.4.1　过拟合现象

过拟合（Over-Fitting）是指针对训练数据，模型过度适配的情况。由于过度的学习模型中的细节和噪声，很容易导致在新的数据上表现较差，这也意味着训练集中的数据噪声被当作某种特征被模型给学习了，从而导致模型的泛化能力变弱。

过拟合现象在无参数非线性的模型中发生的可能性较高，例如在决策树进行训练的过程中，很容易过拟合训练，因此决策树对于解决过拟合问题往往采用剪枝的方式，目的也是移除一些细节对特征的影响。一个非线性模型中过拟合现象大致如图 3-5 所示。

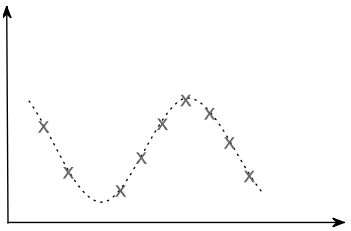


图 3-5　非线性模型中过拟合现象示例

图 3-5 中虚线很好地覆盖到各个点。对于深度神经网络而言，由于它的特征表达能力比较强，因此比较容易产生过拟合的问题，此外大量的参数训练也会导致训练周期加长。

3.4.2　欠拟合现象

欠拟合（Under-Fitting）表示的是模型在训练集和测试集中的表现效果均不佳，本质是获取的数据特征太少，不能有效地拟合数据。欠拟合是模型训练过程中常见的问题，欠拟合相对于过拟合问题，很容易被发现和改进，改进的方法包括：

（1）更换机器学习模型，有可能模式适用的场景与当前场景不匹配。

（2）新增数据的其他特征。新增特征项可以有效避免欠拟合问题。

（3）减少正则化参数。正则化本身是用于解决过拟合问题，但是当模型出现了欠拟合时，可以通过减少正则化参数避免欠拟合问题。

图 3-6 是一个非线性模型的欠拟合现象示例。

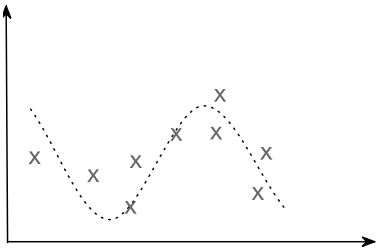


图 3-6　非线性模型的欠拟合现象示例

3.4.3　解决过拟合问题的一般方法

导致过拟合问题的根本原因是特征维度过多，因此解决过拟合问题可以从以下几个方面进行思考。

（1）再次清洗数据。清洗的目的是避免数据不纯导致的过拟合问题。

（2）调整训练集的量。当训练集的量过小时，容易导致学习特征不够集中。对于深度神经网络，一般都需要大量的训练集才能获得相对比较理想的效果。

（3）降低特征维度。可以通过无监督学习筛选特征或者人工干预某些特征。

从工程而言，稀疏规则化是解决过拟合问题的有效方法，常见的稀疏化的方式包括正则化、Dropout 等方法，在实际场景中应用非常广泛。

之所以对数据可以进行稀疏规则化处理，原因是大量的输入特征对于最终的输出结果并不会产生作用或者作用微乎其微，虽然大量的特征可以帮助减少最终模型的训练误差，但是对于新样本进行预测时，反而效果不好，影响了对新样本最终结果的预测。稀疏规则化的引入可以移除这些对结果没有价值的特征，即特征对应的权值为 0。换一个角度，例如，当一个模型如下公式所示时：

$$y = x_1 w_1 + x_2 w_2 + \cdots + x_{1000} w_{1000} + b$$

这是一个 1000 维的信息，但是对于一个结果的影响是否真的需要 1000 维的数据也是值得分析的问题。例如，今天是开车去上班或者乘地铁去上班的影响因素可能是天气或交通情况，不必把微乎其微的是否能够在路上捡到钱包纳入考虑范围。

通过前面的描述，我们知晓稀疏的参数可以防止过拟合，但是由于信息的舍弃总会导致部分特征信息的丢失，因此也可以采用正则化防止过拟合问题。正则化将会保留所有的特征变量，但是会减少特征变量的数量级。

另外一种解决网络结构中过拟合问题的方法是 Dropout。Dropout 是在论文 *ImageNet Classification with Deep Convolutional* 中提到的方法，ImageNet 是根据 WordNet 层次结构组织的图像数据库，其中层次结构的每个节点都由成百上千个图像描绘，目前每个节点平均拥有超过五百张图像。每年都会举办 ImageNet 大规模图像识别挑战赛，属于计算机视觉的顶级赛事。在这些挑战赛中出现了很多优秀的论文和新型的神经网络结构，2017 年是 ImageNet 大规模图像识别挑战赛的最后一届。

在模型训练过程中，某些隐藏层中的权值并不会给整个网络结构带来训练效果， Dropout 是按照一定的概率从神经网络中暂时移除该部分神经元，这种移除只是一种临时移除。在卷积神经网络中可以有效防止过拟合问题。在大规模神经网络的训练过程中，普遍存在两个问题，一是计算周期长，二是容易产生过拟合现象。

过拟合的模型不能真实反映数据的规律，导致模型的有效性降低。Dropout 的应用方式如图 3-7 所示。

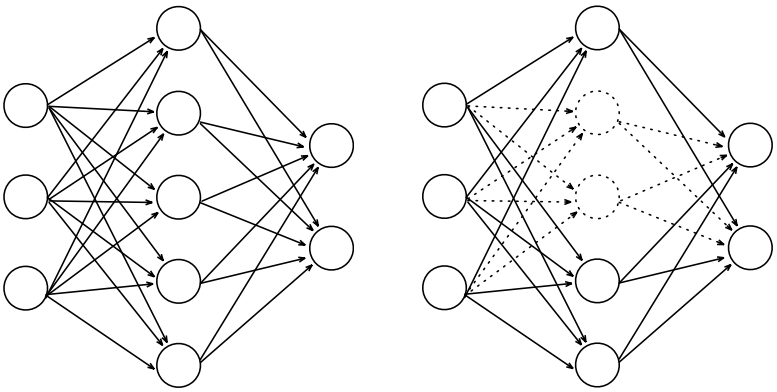


图 3-7　Dropout 的应用方式示例

Dropout 的应用过程如下。

1 随机的从网络中移除一半的隐藏神经元，输入和输出的神经元不做任何变化。

2 对移除之后的网络进行前向传播，根据前向传播的结果计算实际误差，再进行反向传播，通过这样一轮前向传播和反向传播之后，按照随机梯度下降的方式更新当前网络的w和b。

3 恢复第一步被临时移除的隐藏神经元，此时已有部分神经元进行过 w 和 b 的更新，恢复的神经元则一直没变。

不断重复上述步骤，直到训练结果收敛。Dropout 之所以能够通过上述方式解决过拟合问题，原因大致有两方面。

（1）均值作用。Dropout 的过程实质是在讲一个神经网络模型，通过临时移除部分隐藏神经元，构成了一个新的神经网络，不断重复的过程可以理解为在不同神经网络中的不同结果。一般而言，统一的训练数据，经过不同的神经网络，则会得到不同的结果，通过对这些不同的结果进行均值化，使得模型尽可能与真实模型相近。这类平均化的方式可以有效防止过拟合问题。即使不同的神经网络会产生各自的过拟合问题，但是通过综合，会将各自的过拟合问题折中弱化。

（2）避免了神经元之间复杂的共适应关系。由于 Dropout 移除神经元的过程是随机的，因此两个神经元不一定每次都会在 Dropout 之后的神经网络中出现，这意味着训练过程中有时解除了部分神经元之间的相互关联，模型为了达到更好的预测效果，会使得训练过程中神经网络学习更加鲁棒的特征，换个角度即为当模型中某些特征丢失时，模型依然能够进行有效的预测。神经元复杂的共适应关系，会使得模型的鲁棒性降低，最终造成过拟合现象。

3.4.4　实例：拟合与二元一次方程求解

求多元一次方程的解似乎看起来不是一件很难的事情，可以通过方程组的合并求解，从技术可行性上是完全没有问题的，但是能否设计一个通用的模型去求解多元一次方程？例如给定如下方程组：

$$\begin{cases} 5x + 2y = 10 \\ x + 4y = 11 \end{cases}$$

一般情况下，通过方程组的合并消元求解则很容易得$x = 1$、$y = 0.25$。如果写一个求解该二元一次方程的求解函数，似乎相对比较简单，但是如果变量不只是两个，则对方程组进行合并消元的复杂度会加大，甚至会使程序的可读性变差。

机器学习的核心思想源自于数学，方程组合并消元的方法只是数学中人们总结的技巧，但并不是方程组求解的原始思路，我们需要借助最原始的数学思维帮助建立一个数学模型，辅助求解多元一次方程组。对于方程组的求解，实质是约定x和y去拟合两个方程的左右两边相等。对方程组进一步抽象，以矩阵的方式表达，则为：

$$\begin{cases} 5x + 2y = 10 \\ x + 4y = 11 \end{cases} \rightarrow \begin{cases} [5,2,10] \\ [1,4,11] \end{cases}$$

模型应用于求解多元一次方程，拟合过程是使得输入对应的x值能够得到相应的y。后续章节中，会陆续介绍各类神经网络，也是不断在拟合求解。

如果采用感知器求解上述方程，则感知器的训练过程是使得对应的w_i处于某值的时候，输入的x_i能够得到相应的y，实际也是在求w_i的过程。而对多元一次方程求解时，可以将未知数x_i的系数视为w_i，则多元一次方程是已知w_i和y，求x_i的过程。因此，上述方程组可以约定$w_1 = x$、$w_2 = y$，再来求方程的解，即完全转变为感知器的训练过程，求解x和y的过程则为拟合训练w_1和w_2使得方程组成立，而方程式的系数$[5,2,10]$和$[1,4,11]$即为感知器的训练集。

感知器能够解决多元一次方程问题的根本原因也是在于线性可分，理论上其他的机器学习模型也可以做到。如果方程无解，则可以通过最小二乘法，找到一条最佳的直线，尽可能拟合各个点。

3.5　交叉检验

3.5.1　数据类型种类

在机器学习过程中，数据类型大致分为三种：训练数据（Test Data）、验证数据

（Validation Data）、测试数据（Test Data）。训练数据主要用于构建模型。验证数据主要用于模型构建过程中的模型检验和辅助模型的构建，可以适当根据模型在验证集上的效果，对模型进行适当调整。测试数据则是利用数据验证模型的准确性，该部分数据只能在模型测试时使用，用于评估模型的实际有效性，且不能用在模型的构建过程中。当训练数据与测试数据存在交集时，则过拟合问题发生的概率较大。数据类型的构成结构如图 3-8 所示。

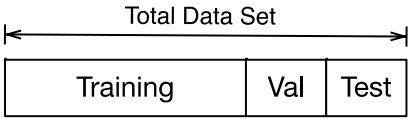


图 3-8　机器学习的数据大致构成

对于上述三者，一般采用 6∶2∶2 的比例分配数据，训练集帮助模型进行参数训练，在验证集上计算验证的误差，优化模型；在测试集上进行误差计算，估算泛化误差，共同保证模型的有效性。验证数据集并不是所有的模型均需要，属于可选的数据集。

可以用高考的例子形象地比喻三者之间的关系，训练数据是平常做题时的各种试卷，通过大量的试卷进行题海战术从而迎接高考，而高考的那次考试即为测试数据，最终确定你的成绩，而高考之前可能会有模拟考试，模拟考试则为验证数据，通过模拟考试发现不足，还可以继续适当修正，模拟考试是可选的，但是平时的练习与最终的高考是必选的。

一般情况下，机器学习模型会将数据集划分为两组：训练数据与测试数据。首先对训练数据进行训练，然后用测试数据进行有效性测试，最终得到模型的测试结果。

上述是最简单也是最容易理解的模型训练、验证、测试方法，但是这种方法至少存在四方面不足。

（1）模型在整个过程中只接受了一次从训练到测试的过程，如果给另外同样量级的训练数据和测试数据，它们之间的结果相差可能会比较大，因此数据的随机性较强。

（2）倘若数据较为稀疏，则测试数据不能很好地代表测试结果。

（3）模型结果很依赖于训练集和测试集的划分方法。

（4）此方法仅用到了部分数据进行模型的训练。一般而言，模型的训练数据越大，体现的特征越多，对于模型的训练结果有一定影响。

因此可以采用其他验证的方式提升测试结果的准确度。

3.5.2　留一交叉验证

留一交叉验证（Leave-One-Out Cross-Validation，LOOCV）的主要思想是：倘若整个数据集为 N，则依次选择一个数据作为测试集，其余 N-1 个数据为训练集，整个过程重复 N 次，保证每一个数据均被作为训练集和测试集，如图 3-9 所示。

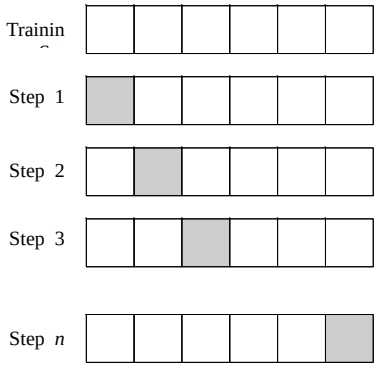


图 3-9　留一交叉验证的示例

对于每一次的训练均会产生一个模型，最终形成 N 个模型。每一次模型对训练数据的测试都会得到一个结果 MSE。对于 N 次模型，则可以采用均值的方式，均值公式如下：

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{MSE}_i$$

留一交叉验证相对于传统的方式，不再受数据集划分的影响，保证了每一个数据都被用作训练集和测试集，充分利用了每一个数据的价值。但是带来的问题也比较明显，对于较大数据量的训练，它的训练需要较长的计算周期。

3.5.3　K 折交叉验证

K折交叉验证是一种训练次数和结果尽可能公正的折中方案，相对于 LOOCV，它的不同点在于测试集不再是单一的一个数据，而是一个集合，具体取决于K的值。例如，当K等于 5 时，表示将所有数据平均划分为 5 份，依次取其中的一份作为测试集，其余 4 份作为训练集，每一次训练之后均会产生一个MSE，将 5 次的MSE进行平均得到最终的MSE，公式如下：

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i$$

实际上，LOOCV 是K折交叉验证的一种特殊情况，即K=N，与 LOOCV 相比，K折交叉验证是一种较好的折中方案。K的选值需要基于模型的偏差和方差进行考虑。如果将模型的准确

度拆解为“准”与“确”，则偏差描述的是“准”，方差描述的是“确”。“准”是模型在预测结果的期望与真实结果的差距，理论上是越小越好；“确”是测试数据在模型上的综合表现，是不同的训练集训练出的模型的实际输出值之间的差异。一般情况下会对两者进行权衡，K的经验值一般选择 5 或 10。

3.6 线性可分与不可分

在利用支持向量机进行实体隐藏关系抽取之前，需要深入了解支持向量机的内部机制。支持向量机是通过构建 N 维（N>1）的超平面对数据分类，这个超平面即是分类边界，如图 3-10 所示。

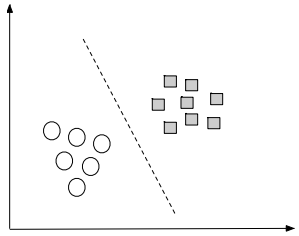


图 3-10　二维空间的超平面对数据分类

图 3-10 所示为线性分类的例子，斜线即是超平面。一般情况下，平面内的超平面的距离可以作为分类预测的准确程度的评判依据。若某一超平面是分类模型中需要的最大间隔值，则这个平面也被称为最大间隔超平面，此时这个分类器也被称为最大间隔分类器，如图 3-11 所示。

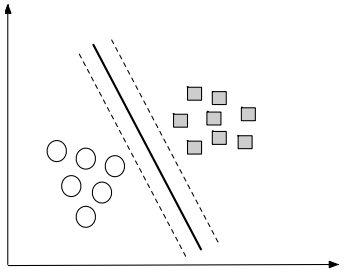


图 3-11　二维空间的最大间隔超平面示例

从分类效果上看，图 3-11 所示的三个超平面中的中间粗体超平面达到的分类效果最佳。图 3-10 和图 3-11 介绍的是线性可分的例子。可以在一个平面中表示其特征，然而一般需要求解的问题常常是线性不可分的，如图 3-12 所示。

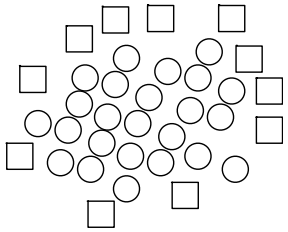


图 3-12　分类器中的简单线性不可分示例

此时，常用的做法是将数据特征映射到高维空间，如图 3-13 所示。

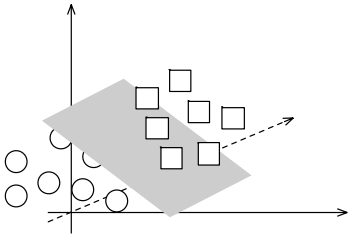


图 3-13　将数据特征映射到高维空间的超平面示例

将线性不可分的数据映射到高维空间会带来一定的问题，维度越高，计算程序越复杂。因此就有了核函数的产生，核函数将特征从低维转换到高维，但是它的计算依然在低维，即使表现形式展现为多维，但避免了在高维上直接进行复杂的数学计算。

3.7　机器学习的学习特征

机器学习是从原始的数据集中抽取出抽象的数据特征，模型的应用效果与数据特征密切相关。一般在进行特征分析时，会从特征的粒度、层次、表示等方面进行。

（1）特征的粒度。特征的粒度是针对特征在原始数据中的大小价值。例如，对于图像，以像素作为图像特征体现不出图像本身的内容。如果一张图像中包含摩托车，以像素点进行特征分析，则很难获得相应的摩托车特征。倘若特征是有意义的局部结构，例如，将车轮、手柄作为特征，则很容易体现摩托车与其他车类型上的差异。

（2）特征的层次。特征的层次则是针对特征在机器学习中的具体形态。例如，在用户画像中，某用户喜欢体育运动，体育运动是用户爱好的特征之一，在体育运动中该用户由衷热爱篮球和足球，其次是羽毛球和乒乓球。对特征进行层次化的表达，可深度表现数据特征，特征表达的层次越深，对数据的理解越透彻。

（3）特征的表示。特征的表示则是对于学习到的特征的表示方式，依然以用户画像为例，

用户的特征是一种树形结构；而对于知识图谱，特征的表示可能是一种图的结构。特征的表示方式可以有效地进行后续的基于特征的行为分析、关联性分析等。

深度学习同传统的机器学习一样，只不过在特征的粒度、层次和表示方面略有不同，尤其在特征的粒度和层次方面，会比传统的机器学习方式更细粒度、更深层次。

3.8 产生式模型与判别式模型

产生式模型（Generative Model）和判别式模型（Discrimitive Model）是机器学习算法中有监督学习的模型概念。对于相同的输入X和输出Y，两者的区别在于产生式模型是估算x和y的联合概率分布$P(x,y)$，而判别式模型估算的是条件概率分布$P(y|x)$。产生式模型可以通过转换得到判别式模型，但是判别式模型不能转换为产生式模型。

对于产生式模型和判别式模型可以通过例子进行阐述，倘若目前有两类水果分别是脐橙和红橙，可以通过两种方法去区分它们：一种是通过分析脐橙、红橙的本身属性以及它们各自拥有的特征，然后通过这些特征识别是属于脐橙还是红橙；另外一种方法是不去分析脐橙和红橙的本身属性，只需要了解它们之间的差异，比如它们在大小或颜色上的差异，通过这些差异去识别脐橙或者红橙。在上述感官的识别过程中，前者即属于产生式模型，后者则属于判别式模型。

用数学的方式，举例而言则是：已知数据(x,y)表示x与y的某种特定关系，对于多个数据：(b,a)、(b,a)、(c,a)、(c,b)，分别通过产生式模型与判别式模型对x值和y值的关系进行分析。

通过产生式模型计算$p(x,y)$，则$p(b,a)=1/2$、$p(b,b)=0$、$p(c,a)=1/4$、$p(c,b)=1/4$。通过判别式模型计算$p(y|x)$，则$p(a|b)=1$、$p(b|b)=0$、$p(a|c)=1/2$、$p(b|c)=1/2$。

产生式模型与判别式模型的优缺点如表 3-2 所示。

表 3-2　产生式模型与判别式模型的优缺点对比

模　　型	优　　点	缺　　点
产生式模型	1．数据信息相对丰富，研究单类别问题灵活性较强 2．能够充分利用先验数据 3．模型可以通过增量学习的方式获得	1．学习过程相对比较复杂 2．在目标分类的问题中容易产生较大的错误率
判别式模型	1．分类边界比较灵活，适用于多类别问题研究 2．能够较好地分辨出类别之间的差异特征	1．不能反映训练样本本身的特性 2．描述信息具备一定局限性

常见的产生式模型包括朴素贝叶斯、隐马尔科夫模型、条件随机场、混合高斯模型等。常见的判别式模型包括支持向量机、逻辑回归、神经网络、最大熵模型等。相对而言，判别式模型的计算性能略优于产生式模型。

3.9　机器学习效果的一般评价指标

一切算法都有其评价的标准，时间复杂度和空间复杂度是算法关于性能的评价标准，但是还需要对算法功能效果进行评估，例如计算结果或预测结果是否符合预期、能否满足需求等。在机器学习算法领域最为广泛且为大众所熟知的评价指标是正确率（Accuracy）、精确率（Precision）及召回率（Recall），基于正确率、精确率和召回率的模型更容易被大众理解和接受，也是最容易执行验证的指标。

假设某个分类器的目标有且仅有两类，分别是正例（Positive）和负例（Negtive），则数据样本中可能存在 TP（True Positives）、FP（False Positives）、FN（False Negatives）、TN（True Negatives）四种状态。TP 表示被分类为正例的个数，即被分类器划分为正例且本身也是正例的个数；FP 表示被错误分类为正例的个数，即被分类器划分为正例但实际上是负例的个数；FN 则表示被错误分类为负例的个数，即被分类器划分为负例但实际上是正例的个数；TN 则表示被正确划分为负例的个数，即被分类器划分为负例且实际上也属于负例的个数。

根据上述四种描述，则所有的正例样本数 P=TP+FN，负例样本数 N=FP+TN，因此各个评价指标如下所示。

（1）正确率。Accuracy=(TP+TN)/(P+N)，即被准确识别为正例和负例的个数比总的样本数量，可以理解为达到期望的结果数量与总数的比。

（2）精确率。Precision=TP/(TP+FP)，即所有识别为正例的个数中真正属于正例的数量比，可以理解为正例的识别正确率。

（3）召回率。召回率是对算法覆盖面的度量，Recall=TP/(TP+FN)，即所有正例样本中，被识别为正例的比率。

仅就上述过程而言，召回率与精确率并无直接关系，但是从实际工程角度，尤其是在大规模数据量中，两者却是相互制约的。以分类器为例，当分类器希望区分出更多正例实例时，也会有很多负例文档被搜索出来，从而精确率受到影响；同理，试图减少负例的实例时，也会使得部分正例的实例被排除在外，从而影响到召回率。召回率越高，则精确率越低；相反，精确率越高则召回率越低，两者的关系如图 3-14 所示。

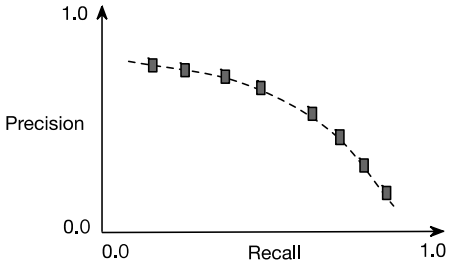


图 3-14 Recall 与 Precision 的关系示例

因此在实际的评价过程中，区分算法效果的优劣不在于精确率或者召回率单一因素。试图将两者的值都提升到高位几乎不可能，因此需要在二者之间寻找平衡的综合因素，F 度量（F-measure）是将召回率和精确率结合的一种方式。F 度量的公式如下：

$$F = \frac{1}{\lambda \times \frac{1}{\text{精确率}} + (1-\lambda) \times \frac{1}{\text{召回率}}}$$

一般情况下取λ=0.5，因此可以将公式简化为：

$$F = \text{精确率} \times \text{召回率} \times 2/(\text{精准率} + \text{召回率})$$

F 值可认为是精确率和召回率的调和平均值。调和平均值与通常的几何平均、算术平均不同，调和平均值强调较小数值的重要性，对小数值比较敏感，因此用于机器学习算法的精确率和召回率比较合适，因为系统试图尽可能让精确率和召回率都比较高。

F 值的计算，可以在精确率和召回率之间得到一定平衡。在所有的机器学习算法中，不仅需要计算一个样本的 F 值，还需要计算更多样本的 F 值，需要对 N 组进行测试并求取平均值，用公式表达如下所示：

$$\bar{F} = \frac{1}{n}\sum_{i=1}^{n} F_i$$

正确率是一个非常直观的评价指标，但是正确率并不一定能够代表一个算法的优劣，尤其在低概率事件中。例如，有一个预测算法，预测某地区是否发生海啸，一般而言发生海啸的概率是极低的，预测的结果只有两个：发生海啸和不发生海啸。如果一个预测算法把所有样本数据不假思索地划分为不发生海啸，则预测算法的正确率可以达到 90%以上，但是当这个地区真正发生海啸的时候，这个预测算法一点效果也没有，形同虚设。在数据不均匀分布的情况下，一味地追求正确率则会忽视算法本身的价值。

在机器学习算法中，除上述评价指标外，还有错误率、鲁棒性、计算复杂度等评价指标。

（1）错误率。错误率与上述的正确率相对，描述在信息处理过程中错误的比率。

（2）鲁棒性。鲁棒性是针对在数据处理过程中，对缺失数据和异常数据的处理能力。一般情况下，鲁棒性越高则说明该算法越智能。

（3）计算复杂度。计算复杂度依赖于具体的实现细节和硬件环境。在做数据分析时，由于操作对象是大量的数据集，因此空间和时间的复杂度问题将是非常重要的一个环节。

（4）模型描述的简洁度。一般而言，模型描述越简洁越受欢迎。例如，采用规则表示的分类器构造法就更加有用。

之所以会有这么多的指标对机器学习算法进行评价，是由于机器学习算法在社会发展的方方面面都会被使用到，从而导致在不同场景下期望不尽相同，需要采用不同的评价指标，引导结果向最初期望的方向发展。例如，错误率和正确率本身表达的含义虽然不相同，但是实质表现的含义却一致，当对一些高精度进行分析时（例如航天工业），则追求错误率极低，在粗粒度的分析（例如天气预报）中追求的则是正确率。

在对测试样本进行正确率测试时，有可能因为过分拟合从而导致测试的结果相对较好，所以更好的测试方法是，在训练集进行构造时，将训练数据分为两部分，一部分用于训练构造器，另一部分用于机器学习模型的效果评估。

3.10　本章小结

本章对机器学习的基础进行了阐述，神经网络和深度学习都是机器学习中的重要组成部分，掌握机器学习的基础内容有助于深入理解神经网络与深度学习。本章对机器学习的类型、常见的函数和重要的参数进行了展开介绍，也阐述了对于过拟合问题、欠拟合问题的一般解决方法。判别式模型和产生式模型是机器学习中的模型定义，对于未来进行模型选择和模型的深入理解有一定帮助。此外，机器学习的效果评估是所有机器学习模型回避不了的问题。对机器学习而言，永远逃不过两类计算：一个是距离计算，另一个是概率计算。

第 4 章

神经网络基础

成年人的大脑中大约有一千亿个神经元。神经网络的本质是通过众多参数以及激活函数去逼真地模拟输入与输出的函数关系，是一种模拟生物的算法模型。除神经网络外，模拟生物的算法还包括蚁群算法、粒子群算法、人工免疫算法、遗传算法。

4.1 概述

神经网络是 20 世纪 80 年代人工智能发展中的热点研究领域。神经网络是一组算法，建模在人脑之后，被设计为识别模式。它们通过机器感知、标记或聚合原始输入来解释感官数据。它们识别的模式是数字，包含在向量中，所有现实世界的数据，无论是图像、声音、文本还是时间序列都必须被翻译。

神经网络可以有效地求解分类问题，它有助于根据示例输入之间的相似性对未标记的数据进行分组。神经网络提取数据的复杂特征，可以将深层神经网络作为涉及强化学习、分类和回归算法的较大机器学习应用的重要组件。

4.1.1 神经网络模型

截至目前，学术界已经提出了各种各样的神经网络模型，例如感知器、前馈型神经网络、卷积神经网络、循环神经网络、组织映射等。这些不同的神经网络模型的差异主要在于神经元的激活规则、神经网络模型的拓扑结构以及参数的学习算法等。

（1）神经元的激活规则：主要针对神经元的输入到输出之间的映射关系，通常是非线性函数，也被称作激活函数。

（2）神经网络模型的拓扑结构：主要是指神经元之间的关联关系，主要包括层数、连接方式（全连接或非全连接）、连接权值等，其中连接权值是神经网络中不断学习和调整的参数。

（3）参数的学习算法：通过训练数据不断训练调整神经网络中的各项参数。

目前常见的神经网络结构包括：前馈网络结构、反馈网络结构和记忆网络结构。

（1）前馈网络。网络模型中的每个神经元根据收到的信息时序分为不同的层。每层的神经元接收上一层神经元的输出，并输出到下一层神经元。 整个网络信息在一个方向传播，没有相反方向的信息传播，是一种有向无环图。 前馈网络结构简单，也非常易于实现。常见的前馈网络包括全连接的前馈型神经网络和卷积神经网络。

（2）反馈网络。反馈网络模型不同于前馈网络，其神经元不仅可以接收来自上层的输出信号，还可以接收自己的反馈信号，因此反馈网络模型在不同的时刻有属于当前不同的状态，具有一定的记忆功能。

（3）记忆网络。记忆网络是在前馈网络和反馈网络的基础之上引入了记忆单元，用于保存神经元计算过程中的中间状态。相对于反馈网络，记忆网络具有更强的记忆能力。

前馈网络、反馈网络、记忆网络的结构大致如图 4-1 所示。

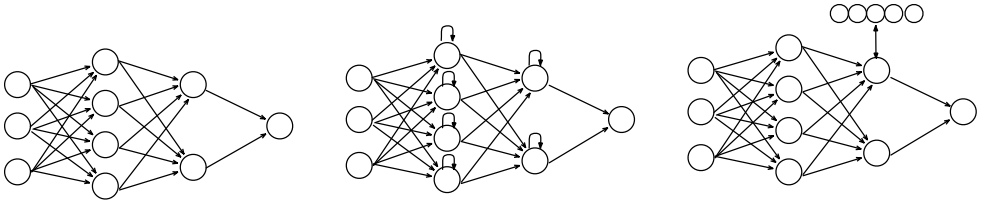


图 4-1　从左至右，前馈网络、反馈网络、记忆网络的结构示例

4.1.2　经典的神经网络结构

常见的多层人工神经网络结构如图 4-2 所示，它由三部分组成：

（1）输入层（Input Layer）。

（2）隐藏层（Hidden Layer）。

（3）输出层（Output Layer）。

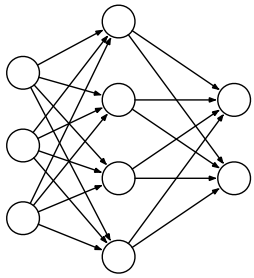


图 4-2　经典的神经网络结构组成示例

由于神经网络的结构类型在不断变异，因此上述的组成结构并不适用于所有的神经网络。

4.1.3　一般业务场景中神经网络适应性

各类神经网络适用的场景如表 4-1 所示。

表 4-1　各类神经网络适用的场景示例

数据类型	业务案例	输　　入	变　　换	采用的神经网络
文档	主题模型 语义哈希	词频概率	可为二进制	深度自动编码器
图像	图像识别	二进制	二进制（可见层及隐藏层）	深度置信网络
图像	图像识别	连续	高斯修正	深度置信网络
图像	多对象识别	N/A	高斯修正	卷积网络、递归张量神经网络
图像	图像搜索 语义哈希	N/A	高斯修正	深度自动编码器
音频	语音识别	N/A	高斯修正	循环网络
音频	语音识别	N/A	高斯修正	移动窗口、递归张量神经网络或卷积网络
时间序列	预测分析	N/A	高斯修正	循环网络
文本	情感分析	词向量	高斯修正	递归张量神经网络或深度信念网络
文本	命名实体识别	词向量	高斯修正	递归张量神经网络或深度信念网络
文本	词性标注	词向量	高斯修正	递归张量神经网络或深度信念网络

续表

数据类型	业务案例	输　　入	变　　换	采用的神经网络
文本	语义角色标记	词向量	高斯修正	递归张量神经网络或深度信念网络

4.1.4　神经网络的深度

这个“深”与神经网络连接的复杂性非常接近。对于神经网络的复杂性可以采用层数、神经元数量或连接权数作为度量。相比之下，数据本身的复杂性，可以采用使用数据的比例与标签和非标记数据的比例来衡量。

一般情况下，增加训练的复杂性总是能够适应样本，并且获得良好的泛化能力，但这取决于数据规模的大小和更抽象的特征。如果只有单一特征，则不需要使用深度神经网络来训练。

神经网络的深度主要体现在以下两个方面。

（1）网络层级的深度。对于一般神经网络而言，深度主要表现在隐藏层的数量，理论上而言，网络层次的深度越深，神经网络能够拟合的能力越强，越能够解决更复杂的问题。

（2）特征级的深度。深层次表达特征是能够解决复杂问题的重要基础之一，越是复杂的问题，对于特征深度的挖掘越深，更细粒度的特征，可以有效地对样本数据加以区分。

特征本身可以构成一个复杂的网络结构，较低层次的特征比较容易确定，而高层次的特征则由底层特征组合形成，这些特征并不是很容易抽象出来。例如，如果以“聪明”作为孩子的特征，那么“聪明”这个特征则是由更多的子特征组成的。同时，神经网络的深度决定了计算的复杂度。

4.2　常见学习方法

4.2.1　误差修正学习

误差修正学习也叫 Delta 学习规则。误差修正学习通常与监督学习一起使用，是将系统输出与期望输出值进行比较，并使用该误差来改进培训。最直接的方法是可以使用误差值来调整权值，使用诸如反向传播算法的方式。如果系统输出为y，并且所需的系统输出已知为d，则误差值可定义为：$e = d - y$。

误差修正学习方法尝试在每次训练迭代时最小化该误差信号。采用误差修正学习的最流行的学习算法是反向传播算法。

神经元的作用是基于输入向量$\boldsymbol{X}=(x_1,x_2,x_3,\ldots,x_n)$产生相应输出$y$。为了使得神经元的输出符合预期，则需要训练它。训练样本是一系列已知的 x 和$\hat{y}$数据集，其中$\hat{y}$表示预期的正确输出。可以采用如下公式来描述实际输出y和预期的正确输出$\hat{y}$之间的误差：

$$E=\frac{1}{2}(y-\hat{y})^2$$

采用上式是为了计算的方便，如定义为$E=|y-\hat{y}|$也没有问题。学习训练的过程是减少错误E，直至趋近于 0 为最佳。 神经元的错误通过迭代得到最小值，每次根据当前情况进行一点修正，逐渐找到最小目标函数，这与生物神经元逐渐生长的策略有一定相似度。梯度下降法对于找到这样的最小值有很大的帮助。例如，当寻找一个山谷的最低海拔位置时，应该试着向下坡方向前行，当然前提是它是一个单一山谷，没有复杂的起伏。如果它是一个起伏不断的山谷，则很容易在一个局部的小山谷中错误地认为达到了最低海拔点。对于单个神经元，误差最小化优化问题的目标函数是一个凸函数。

4.2.2 赫布学习规则

赫布理论（Hebbian Theory）是一种神经科学理论，提出了在学习过程中对大脑神经元适应性的解释，描述了突触可塑性的基本机制，其中突触功效的增加来源于突触前细胞重复和持续刺激突触后细胞。唐纳德·赫布（Donald Hebb）在 1949 年出版的《行为的组织》（*The Organization of Behavior*）中引入了这一理论。

赫布学习规则是最简单、最传统的神经元学习规则。从人工神经元和神经网络的角度来看，赫布学习规则的原理可以被描述为一种确定如何改变模型神经元之间权值的方法。如果两个神经元同时被激活，则这两个神经元之间的权值就会增加；如果它们分别被激活，则它们之间的权值会降低。倾向于同时为正或为负的节点具有较强的正权值，而倾向于相反的那些节点具有较强的负权值，这点与“条件反射”有一定的相似性，赫布学习规则代表一种纯前馈、无监督学习。

一个简单的赫布学习公式：$w_{ij}=x_ix_j$，其中w_{ij}表示神经元j到神经元i的连接权值，x_i表示神经元i的输入。值得说明的是，一般而言，当i等于j时，w_{ij}的值恒为 0。而对于调节的Δw_{ij}，则为$\Delta w_{ij}=w_{ij}(n+1)-w_{ij}(n)=\eta y_ix_j$，$w_{ij}(n+1)$表示已经进行$n+1$次调整权值之后从节点$j$到节点$i$的连接权值，$\eta$是学习速率，$x_j$为节点$j$的输出并作为节点$i$的输入，$y_i$为节点$i$的输出。对于 Hebb 学习规则，神经元的输出值可以用$\text{fun}(W^{\mathrm{T}}X)$表示，因此$\Delta W=\eta\text{fun}(W^{\mathrm{T}}X)X$，例如，如图 4-3 所示的简单的神经网络。

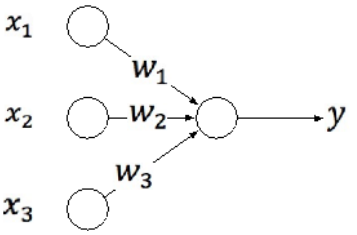


图 4-3　简单的三个输入向量的神经网络示例

根据图 4-3 所示，输入向量为$\boldsymbol{X}=[x_1,x_2,x_3]^{\mathrm{T}}$，权值向量$\boldsymbol{W}=[w_1,w_2,w_3]^{\mathrm{T}}$，设定初始权值向量的值为$\boldsymbol{W_0}=[1,0.5,-1]^{\mathrm{T}}$，训练数据三组输入向量$\boldsymbol{X_1}=[1,0.5,1]^{\mathrm{T}}$、$\boldsymbol{X_2}=[-2,1,2]^{\mathrm{T}}$、$\boldsymbol{X_2}=[-3,0,-0.5]^{\mathrm{T}}$，对应的输出$\boldsymbol{Y}=[1,0,-1]^{\mathrm{T}}$，设定$\eta$为 1，同时激活函数约定为阶跃函数 sgn。而已知$\boldsymbol{W_1}=\boldsymbol{W_0}+\Delta\boldsymbol{W}$，因此需要首先计算出$\Delta W$，对于训练数据的第一组输入向量$\boldsymbol{X_1}$：

$$\Delta\boldsymbol{W_1}=\eta fun(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X})\boldsymbol{X}=1\times\mathrm{sgn}(\boldsymbol{W}_0^{\mathrm{T}}\boldsymbol{X_1})\boldsymbol{X_1}$$

通过上述计算即可得到第一组训练的输入$\boldsymbol{X_1}$对应的权值调整，得到新的$\boldsymbol{W_1}$之后，继续进行输入向量$\boldsymbol{X_2}$和$\boldsymbol{X_3}$调整权值即可。

4.2.3　最小均方规则

最小均方规则（Least Mean Square，LMS）也被称作 Widrow-Hoff 学习规则，是 1962 年由 Bernard Widrow 与 Marcian Hoff 提出的学习规则。基本原理是对一组包含n个元素的$\{x_1,x_2,x_3,\dots,x_n\}$的输入作为一个线性组合加权求和，对得到的求和结果$y$与期望输出$d$进行比较，计算误差值$e$，并根据误差值的大小对权值进行调整。

例如，设定对于n个输入元素$\{x_1,x_2,x_3,\dots,x_n\}$都有$n$个对应权值向量$\{w_1,w_2,w_3,\dots,w_n\}$，则加权求和的$y$可以按照如下公式计算：

$$y=x_1w_1+x_2w_2+x_3w_3\dots+x_nw_n=\sum_{i=0}^{n}x_iw_i$$

如果以向量的方式表示则为：$y=\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}$。

实际输出 y 与期望输出值d的差值为e，公式如下：

$$e=d-\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}$$

因此权向量的调整向量值可以按照如下公式计算：

$$\Delta\boldsymbol{W}=\eta(d-\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X})\boldsymbol{X}$$

对于一个具体的W，则计算公式为：$\Delta w_i=\eta(d-\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X})x_i$。

最小均方规则是一种非常简单易懂的学习规则，它的学习方式与神经元的激活函数没有任何关系，也不需要对激活函数求导，不仅学习速度比较快，而且有一定的计算精度。权值的初始化也比较自由，最小均方学习规则实则是误差修正学习的一种特例，适用于比较简单的模型训练，最小均方规则学习方法常常用于线性回归问题的求解。

4.2.4 竞争学习规则

竞争学习是神经网络中无监督学习的一种形式，通过竞争学习增加网络中每个节点的专业性。竞争学习规则的原理是，神经网络的输出神经元相互之间产生竞争以激活自身，结果在某一时刻仅会有一个输出的神经元被激活，这个被激活的神经元被称作竞争胜利的神经元，其状态会被激活，其他失败的神经元状态将会被抑制。

实验表明，一个兴奋的神经元细胞会对周围神经元的神经细胞产生抑制作用，使得神经细胞之间出现竞争，也可能促使多个细胞兴奋，但一个兴奋强度较高的的神经细胞对周围兴奋程度较低的神经细胞的抑制作用也越强，其结果使得周围神经元细胞整体兴奋程度减弱，从而兴奋程度最强的神经元细胞在这次竞争中胜出，而其他神经元失败。这类抑制性作用通常满足一定的函数分布关系，如距离越远则抑制作用越弱，距离越近则抑制作用越强。

竞争学习采用“胜者为王”的学习策略，主要分为三个步骤。

1. 向量的归一化。由于不同的模式之间单位不一定相同，因此在进行数据处理前，会将模式向量$\boldsymbol{X}$进行统一方式和大小处理，使得模式之间具备比较性。因此需要将神经网络中的输入模式向量$\boldsymbol{X}$和竞争层中各神经元对应的内星权向量$\boldsymbol{w_j}$全部进行归一化处理，处理公式如下：

$$\hat{X} = \frac{X}{||X||} = \{\frac{x_1}{\sqrt{\sum_{i=1}^{n} x_i^2}}, \cdots, \frac{x_n}{\sqrt{\sum_{i=1}^{n} x_i^2}}\}$$

2. 寻找获胜的神经元。通过神经网络输入一个输入模式向量$\boldsymbol{X}$时，让竞争层的所有神经元对应的内星权向量与输入模型向量$\boldsymbol{X}$进行相似性比较，将与$\boldsymbol{X}$最相似的内星权向量判为竞争获胜神经元，其权向量记为$\boldsymbol{w_j}$，相似性的计算可以采用欧氏距离或余弦法等。

3. 权值调整。“胜者为王”的学习策略约定，获胜的神经元输出为 1，其余神经元输出为 0。只有获胜神经元才有资格调整其权向量，调整权向量的公式如下，其中w_{kj}表示连接输入节点j和神经元k的突触权值，η表示学习速率，不断迭代调整最终使得网络收敛。

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_{kj}), & \text{神经元}k\text{赢得竞争} \\ 0, & \text{神经元}k\text{竞争失败} \end{cases}$$

除此之外，竞争学习规则有三个基本要素。

（1）除一些随机分布的突触权值外，一组神经元都是相同的，因此对给定的一组输入模式做出不同的响应。

（2）对每个神经元的权值施加限制。

（3）允许神经元竞争权力来响应给定的输入子集的机制，使得每次只有一个输出神经元（或每个组仅有一个神经元）被激活。

因此，网络的各个神经元学习专门从事类似模式的集合，并且这样做成为不同类型的输入模式的“特征检测器”。

竞争网络将相关输入集合到少数输出神经元之一的事实基本消除了代表性的冗余，这是生物感觉系统中处理的重要部分。当接收到更多的数据时，每个节点收敛到集群的中心，并且对于该集群中的输入更强烈地激活，因此对于其他集群中的输入而言更弱。

4.2.5　其他学习规则

除上述经常接触的学习规则外，还包括玻尔兹曼学习规则、外星学习规则等。

（1）玻尔兹曼学习规则。玻尔兹曼学习规则被推导出一种根植于随机学习算法的统计力学的思想。基于玻尔兹曼学习规则的神经网络常被称作玻尔兹曼机。玻尔兹曼机的神经元与其他神经网络的神经元有一定差异，它仅有激活的和抑制的两个状态，即 1 或 0。层与层之间的权值可以用一个$|V|\times|H|$大小矩阵表示。在网络参数更新时，算法难点主要在于对权值和偏置项的求导。由于 V 和 H 中的值为二值化的值，不存在连续且可导的函数进行计算，因此在实际计算过程中，常常借助 Gibbs 采样方法，而玻尔兹曼机的提出者 Hinton 在其中采用了对比分歧的方法来更新模型参数。

（2）外星学习规则。在神经网络中可以将节点划分为两类节点。一类可以划分为内星节点，内星节点的特征是总是接收来自四面八方的输入加权信号，内星节点是信号的汇聚点，对应的权值向量可以被称为内星权向量。另一类可以划分外星节点，与内星节点相反，外星节点的特征是总是向四面八方发出输出加权信号，属于信号的发散点，对应的权值向量可以被称为外星权向量。外星学习规则属于有监督的学习方式，目的是为生成一个期望的输出两项，其学习规则可以用公式$\Delta \boldsymbol{W}_j = \eta(d - \boldsymbol{W}_j)$表示，含义为尽量使节点$j$对应的外星权向量向期望的输出$d$靠近。

无论采用哪一种学习规则，其核心价值是一致的，通过调整各类参数，使得期望输出与实际输出尽可能相似。

4.3 优化方法：梯度下降

4.3.1 概述

优化算法的作用是通过不断改进模型中的参数使得模型的损失最小或准确度更高。在神经网络中，训练的模型参数主要是内部参数，包括权值（W）和偏置（B）。模型的内部参数在有效训练模型和产生准确结果方面起着非常重要的作用。 因此需要使用各种优化策略和算法来更新和计算影响模型训练和模型输出的网络参数来近似或达到最优值。常见的优化算法分为两类。

（1）一阶优化算法。

该算法使用参数的梯度值来最小化损失值。最常用的一阶优化算法是梯度下降。函数梯度可以采用导数$\frac{dy}{dx}$的多变量表达式进行表达，用于表示y相对于x的瞬时变化率。通常为了计算多变量函数的导数，用梯度代替导数，并使用导数来计算梯度。梯度和导数之间的主要区别在于函数的梯度形成一个向量场。

（2）二阶优化算法。

二阶优化算法使用二阶导数（也称为 Hessian 方法）来最小化损失值，以 Newton 法和 Quasi-Newton 法为代表的二阶优化算法目前最大的困难在于计算复杂度，正是由于二阶导数的计算成本较高，因此该方法未得到广泛应用。

4.3.2 梯度下降法

梯度下降法（Gradient Descent）是机器学习中最常用的优化方法，常常用于求解目标函数的极值。梯度是一个向量，表示函数在该点处的方向导数沿着该方向取得最大值，即函数在该点处沿着该方向变化最快、变化率最大，这个方向即为此梯度的方向，变化率即为该梯度的模。梯度下降是一种基于不断迭代的运算方法，每一步都是在求解目标函数的梯度向量，在当前位置的负梯度方向作为新的搜索方向，从而不断迭代。之所以采用在当前位置上的梯度反方向，是由于在该方向上的目标函数下降最快，可以找到局部最小值；同理，要是沿着梯度的正方向作为新的搜索方向，则找到的是局部最大值。

例如，对于输入(x_1, x_2)分别表示人的身高和体重，通过$f(x_1, x_2)$输出的是偏胖或不偏胖的结果。现给定一堆的(x_1, x_2)集合，通过训练使得$f(x_1, x_2)$对于输入的值都能够判定其属于偏胖或不偏胖。拟合函数可以采用如下公式：

$$f(x) = \theta_1 x_1 + \theta_2 x_2 + \theta_3$$

训练的过程则是找到有效的 $f(x)$函数，使得训练中的样本均满足 $f(x)$的输出结果。因此可以定义一个误差函数如下所示，误差函数表达的是预测值与真实值差的平方和的一半：

$$\text{Loss}(\theta)=\frac{1}{2}\sum_{i=1}^{n}[f(x_i)-y_i]^2$$

其中x_i表示第 i 个输入样本，y_i表示训练样本的期望输出，$f(x_i)$是实际训练结果的输出。对于整个系统而言，误差函数越小，则表示对训练样本的拟合程度越高，因此可以将问题转换为对$\text{Loss}(\theta)$求解极小值（极小值时误差最小），换言之，则为当θ_1、θ_2、θ_3取何值时，整个系统的误差最小。因此需要求解$\text{Loss}(\theta)$的梯度，即依次对θ_1、θ_2、θ_3进行求偏导数，如下推导公式：

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\text{Loss}(\theta)&=\frac{\partial}{\partial\theta_j}\frac{1}{2}(f(x)-y)^2\\&=2\times\frac{1}{2}(f(x)-y)\times\frac{\partial}{\partial\theta_j}(f(x)-y)\\&=(f(x)-y)\times\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i-y\right)\\&=(f(x)-y)x_i\end{aligned}$$

在求得偏导数之后，θ 需要在当前梯度位置的反方向调整，公式如下：

$$\theta_j=\theta_j-\eta\times\frac{\partial\text{Loss}}{\partial\theta_j}$$

其中η是学习速率。经过多次迭代即可得到最优的θ_1、θ_2、θ_3，即在这些参数之下，$f(x)$训练的样本整体误差值最小。

常用的梯度下降法还包含三种不同的形式，分别是批量梯度下降法（Batch Gradient Descent）、随机梯度下降法（Stochastic Gradient Descent）和小批量梯度下降法（Mini-batch Gradient Descent）。

（1）批量梯度下降法。批量梯度下降法是每次使用全量的训练集样本来更新模型参数，整个过程简单易于实现。对于凸目标函数获得的是全局最优解，对于非凸目标函数可以保证一个局部最优解。正是由于批量梯度下降每次更新都会使用所有训练数据，因此训练过程比较慢，尤其在训练数据较大时。

（2）随机梯度下降法。随机梯度下降法是每次从训练集中随机选择一个样本来进行学习，训练的速度比较快，每次迭代的计算量也不大，但是训练速度较快会牺牲一部分准确度，并且

最终计算的结果并不一定是全局最优解，整个过程的实现相对较为复杂，迭代次数也相对较多。同时，随机梯度下降法的随机性可以有一定可能性跳出局部最优解，也可能导致收敛的复杂化，即使训练已经达到了最优，也会进行过度的优化。

（3）小批量梯度下降法。小批量梯度下降法综合了批量梯度下降和随机梯度下降两种方法，每次更新从训练集中随机选择 m（$m<n$）个样本进行学习，收敛过程相对稳定可靠。

目前，梯度下降法是深度学习中最为常用的优化算法，使用梯度下降法及其变体也面临一些问题，例如选择正确的学习速率相对较为困难。太小的学习速率会导致网络收敛速度太慢，太大的学习速率会使学习过程处于一个不稳定的环境，导致结果缺乏稳定性。不仅如此，相同的学习速率不适合所有参数更新，对于稀有特征，似乎增加其学习速率更有利于训练。

4.3.3 梯度下降的优化算法

一般情况下，在采用小批量梯度下降法时会将训练集分为 n 个批量集（batch），每一个批量包含 m 个样本数据，每次都对一个批量进行训练，然后根据当前批量训练的结果更新参数，公式化表示为$w_{t+1} = w_t + \Delta w_t$，而$\Delta w_t = -\eta g_t$，其中$\eta$为学习速率，$g_t$表示$w$在$t$时刻的梯度。

采用小批量梯度下降法，可以减少训练的计算量，并在较快的时间内达到收敛，但是这种方式非常依赖于当前训练的批量样本，会导致更新存在一定的不稳定性。可以采用下面这些方法对梯度下降法再次进行优化。

（1）动量（Momentum）法。动量法模拟的是物体运动过程中的惯性，在参数更新的过程中，需要一定时间保持在原有方向上的趋势，但同时也会利用当前批量数据的训练情况对参数进行微调更新，用公式表示为$\Delta w_t = \mu w_{t-1} - \eta g_t$，其中$\mu$即为动量参数。从公式上看，在调整参数时，保持了一定的惯性，学习速度更快且这种惯性对于解决局部最优有一定帮助。μ值的大小，表示在多大程度上保持原有的参数更新方向，一般取值(0,1)，默认一般设为 0.9，也可以在不同的训练状态中调整μ值的大小。例如，在训练初始阶段，梯度的变化相对较大，μ值可以选择适当小的值，相反，到训练后期，可以选择较大的值。

在动量法的基础之上，还衍生出了 Nesterov 动量法。

（2）AdaGrad 方法。在很多情况下，对所有参数的更新基本都是采用相同的更新速率，但这样的方式并不合适。例如，有些参数希望更新范围能更大一点，而有些参数仅仅需要微调。Adagrad 方法就是针对此问题的解决方法，它为各个参数适配了不同的学习速率，公式如下所示：

$$\Delta w_t = -\frac{\eta}{\sqrt{\sum_{k=1}^{t} g_k + \epsilon}} g_t$$

其中g_t表示当前t时刻的梯度，η是默认的学习速率，ϵ为非零常数，通过上述公式不难理解对于每一个参数，随着更新距离之和的增加，其学习速率也越来越小，这也符合在模型训练的初期，希望参数的变化更多更快；而在模型训练的后期，希望参数变化得更慢且值更小。

在 AdaGrad 基础之上，衍生出了 AdaDelta 方法。主要用来解决 AdaGrad 方法的不足，例如，由于学习速率是单调递减的，因此在训练的后期学习会速率非常小。

（3）采用 RMSProp 方法。RMSProp 方法介于 AdaGrad 方法和 AdaDelta 方法之间，通过增加衰减系数来控制早期信息对当前的影响，从而避免 AdaGrad 方法中学习速率趋于零的问题，公式如下所示：

$$\Delta w_t = -\frac{g_t}{\sqrt{E|{g_t}^2| + \epsilon}} \times \eta$$

其中，g_t表示当前t时刻的梯度，η是默认的学习速率，ϵ为非零常数，t时刻的期望$E|{g_t}^2|$可通过$t-1$时刻的期望$E|{g_{t-1}}^2|$和当前的梯度加权求和得到，具体计算公式如下：

$$E|{g_t}^2| = \mu \times E|{g_{t-1}}^2| + (1-\mu) \times {g_t}^2$$

其中μ为 0 到 1 之间的实数。

（4）采用 Adam 方法。Adam 方法利用梯度的一阶矩估计和二阶矩估计为各个参数适配不同的学习速率，公式如下所示：

$$\Delta w_t = -\frac{\widehat{m_t}}{\sqrt{\widehat{n_t}} + \epsilon} \times \eta$$

其中，$\widehat{m_t}$表示期望$E|g_t|$的无偏估计，$\widehat{n_t}$表示期望$E|{g_t}^2|$的无偏估计，g_t表示当前t时刻的梯度，η是默认的学习速率，ϵ为非零常数。$\widehat{m_t}$和$\widehat{n_t}$可通过如下公式计算：

$$\widehat{m_t} = \frac{m_t}{1-\mu^t}$$

$$\widehat{n_t} = \frac{n_t}{1-v^t}$$

上式中m_t、n_t分别为g_t的一阶矩估计和二阶矩估计，μ和v均为 0 到 1 之间的实数。m_t和n_t可通过如下公式求得：

$$m_t = \mu \times m_{t-1} + (1-\mu) \times g_t$$

$$n_t = v \times n_{t-1} + (1-v) \times {g_t}^2$$

Adam 方法通过梯度的矩估计动态调整了学习速率，同时将学习速率限制在明确的范围内，使得参数变化比较平稳。Adam 方法结合了 AdaGrad 方法和 RMSProp 方法的特点，适用于稀

疏梯度，以及非平稳目标的场景。

上述梯度下降优化算法都是为了更有效地对模型进行训练。在实践中，采用 Adam 方法得到的效果相对优于其他方法。与其他自适应优化学习算法相比，Adam 方法收敛速度更快，学习效果更有效，可以纠正其他优化技术中的问题，例如学习率消失、收敛太慢等问题。

4.3.4 梯度消失问题

梯度消失问题（Gradient Vanishing Problem）在神经网络层数相对较多时可能会遇到，且随着神经网络层数的不断增加，发生的概率越明显。对于梯度消失问题，可以通过图 4-4 进行说明。

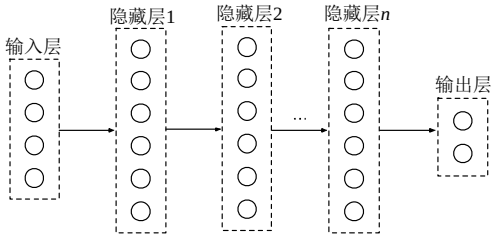


图 4-4 多隐藏层的神经网络示例

如图 4-4 所示，针对包含n层的隐藏层的神经网络，一般情况下第n个隐藏层的权值还可以正常更新，但是相对于第 1 层和第 2 层的隐藏层，可能存在层的权值更新几乎不变，基本和初始的权值相差较小，这个时候则是发生了梯度消失问题。尤其在n较大时，发生梯度消失的概率更高，此刻第 1 层和第 2 层隐藏层只是被当作了一个映射层，并没有发挥其层次结构的价值。

梯度消失的原因在于激活函数与层次结构的选择，假设存在如图 4-5 所示的单一神经网络结构，每一层均只有一个神经元。

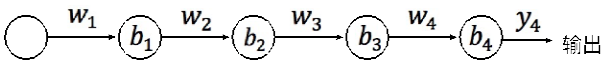


图 4-5 单一的神经网络结构示例

假设图 4-5 表示的神经网络的定义为$y_i = \text{sigmod}(z_i)$，且$z_i = w_i x_i + b_i$，则可以推导梯度的表达式：

$$\frac{\partial \text{output}}{\partial b_1} = \frac{\partial \text{output}}{\partial y_4}\frac{\partial y_4}{\partial z_4}\frac{\partial z_4}{\partial x_4}\frac{\partial x_4}{\partial z_3}\frac{\partial z_3}{\partial x_3}\frac{\partial x_3}{\partial z_2}\frac{\partial z_2}{\partial x_2}\frac{\partial x_2}{\partial z_1}\frac{\partial z_1}{\partial b_1}$$
$$= \frac{\partial \text{output}}{\partial y_4}\text{sigmoid}'(z_4)w_4\text{sigmoid}'(z_3)w_3\text{sigmoid}'(z_2)w_2\text{sigmoid}'(z_1)$$

而激活函数$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$，因此其导数形式$\text{sigmoid}'(x)$的函数为$\text{sigmoid}'(x) =$

$x(1-x)$，根据$\text{sigmoid}'(x)$的表达式，可以知道$\text{sigmoid}'(z_i)$的最大值为 0.25，而w_i本身的值在(0,1)之间，因此$\frac{\partial \text{output}}{\partial b_1}$的值随着神经网络层次的加深，值越来越小直至消失，因而发生梯度消失的现象。

归纳而言，由于在反向传播算法过程中，使用的是矩阵求导的链式法则，即通过一系列的连乘，且连乘的数字在每层都小于 1，因此对于梯度越来越小，最终导致梯度消失。

从梯度消失的角度而言，可以说明并不是网络结构设计得深度越深越好，但是理论上网络结构的深度越深，表达能力和抽象能力就越强，因而对于梯度消失问题，需要从激活函数以及网络层次结构设计上着手解决，例如激活函数可以考虑用 ReLU 函数代替 Sigmoid 函数。

与梯度消失问题相反的则是梯度爆炸（Exploding Gradient），如果在反向传播算法过程中，连乘的数字在每层都大于 1，则梯度会越来越大，从而导致梯度爆炸问题的发生。梯度爆炸问题相对容易解决，可以通过简单地设置阈值来避免梯度爆炸。

4.3.5　示例：利用梯度下降法求函数极值

利用梯度下降法求$y=x^2$的函数极值，采用学习速率为 0.4，x的计算起始位置从-3 开始，目标函数是$y=x^2$，可以得到其导数为$f'(x)=2x$，梯度下降的迭代效果如表 4-2 所示。

表 4-2　梯度下降的迭代效果示例

迭代次数	x 所在位置	调整方向	新的 x 值位置
1	-3	-f'(-3)×0.4	-3+(-f'(-3)×0.4)=-6.00000000e-01
2	-6.00000000e-01	-f'(-6.00000000e-01)×0.4	-6.00000000e-01+(-f'(-6.00000000e-01)×0.4)=-1.20000000e-01
3	-1.20000000e-01	-f'(-1.20000000e-01)×0.4	-1.20000000e-01+(-f'(-1.20000000e-01)×0.4)=-2.40000000e-02
4	-2.40000000e-02	-f'(-2.40000000e-02)×0.4	-2.40000000e-02+(-f'(-2.40000000e-02)×0.4)=--4.80000000e-03
...	...	...	...

x在迭代过程中，不断趋近于$f(x)$的极值所在位置 0，通过不断迭代，收敛之后，获得相应的x值，即可求出$f(x)$的极值。

从上述示例中，不难看出在梯度下降计算过程中，其特点是：随着目标值的不断接近，步长越来越小，下降的速度也逐步减慢。

分别对学习速率采用 0.1、0.2、0.4、0.8，进行$y=x\times x$的极值求解，迭代起始的x为 10，迭代次数为 5，计算收敛情况如图 4-6 所示。

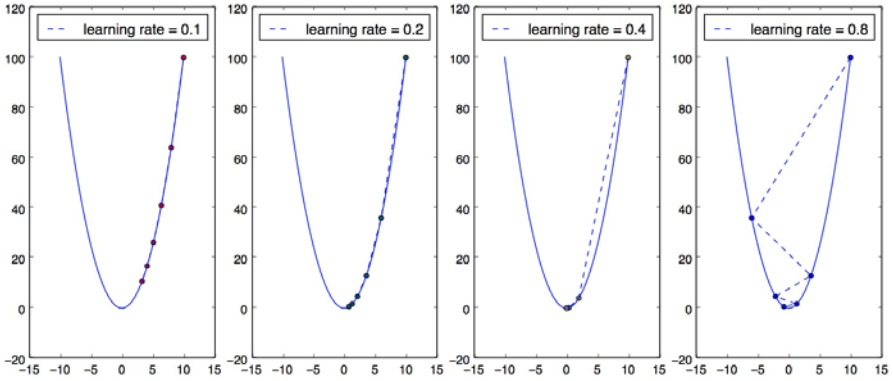


图 4-6　不同学习速率下的迭代收敛情况示例

通过图 4-6 可以发现，当学习速率较小时，不能快速地收敛到相应的位置；当学习速率较大时，则不能很好地收敛到相应位置，即便数据有抖动。

4.4　常见的神经网络类型

4.4.1　前馈型神经网络

前馈型神经网络（Feedforward Neural Network）是人工智能领域中，最基本和最简单的神经网络类型，常见的前馈型神经网络包括：感知器网络、BP 神经网络和 RBF 网络。目前在理论和实际应用中都达到了较高的水平。

前馈型神经网络中，各个神经元分别处于不同的层次结构，但是每一层的神经元都可以接收上一层神经元的输出信号，并向下一层神经元传递信号。在模型的内部结构中，参数从输入层逐步向输出层传播，并且传播过程中是单向传播的。常见的前馈型神经网络有感知器网络、反向传播神经网络（BP 神经网络）和径向基函数神经网络（RBF 神经网络）。

（1）感知器网络。感知器也被称作感知机，它是最简单的前馈型神经网络，主要用于模式分类，也可以用在学习控制和基于模式分类的多模态控制。感知器网络可以分为单层感知器网络和多层感知器网络。

（2）反向传播神经网络。BP 神经网络利用了权值的反向传播调整策略，与感知器不同的是，BP 神经元网络传递函数基于 Sigmoid 函数，所以输出的值范围是[0,1]，可以实现从输入到输出的任意非线性映射。

（3）径向基函数神经网络。RBF 网络能逼近任意的非线性函数，可以处理难以解析的规

律性问题，具有良好的泛化能力和快速收敛速度，成功地应用于非线性函数逼近、时序分析、分类问题、模式识别、信号处理、图像处理、系统建模、控制和故障诊断等方面。

4.4.2　反馈型神经网络

反馈型神经网络（Feedback Neural Network）是一种反馈动力学系统，也被称作回归网络。在反馈型神经网络模型中，网络的输入信号决定了整个反馈系统的初始状态，网络模型会通过一系列的状态转换逐渐达到内部的收敛平衡状态，这种平衡状态即为反馈型神经网络的计算输出，平衡状态是网络模型稳定性的表现。在反馈型神经网络模型中，稳定性是非常重要的特征。目前常见的传统反馈型神经网络主要包括 Hopfield 神经网络、Elman 神经网络，以及海明神经网络、双向联想存储器网络等。

（1）Hopfield 神经网络。Hopfield 神经网络是于 1982 年由生物物理学家 J.Hopfield 等提出的典型的反馈型神经网络，并利用 Hopfield 神经网络成功地探讨了旅行商问题的求解方法。网络结构中从输出到输入有反馈连接。Hopfield 神经网络分为离散型 Hopfield 神经网络和连续型 Hopfield 神经网络。

（2）Elman 神经网络。Elman 神经网络是于 1990 年由 J. L. Elman 首先提出的针对于语音处理问题的神经网络模型，它是一种典型的局部回归网络。Elman 神经网络可以看成是一个具有局部记忆单元和局部反馈连接的反馈型神经网络，除传统输入层、隐藏层和输出层外，还包含了一种新的层次结构——关联层。

值得说明的是，深度学习中的递归神经网络和循环神经网络都属于反馈型神经网络。

4.4.3　自组织竞争型神经网络

前馈型神经网络和反馈型神经网络基本都是一种有监督的神经网络模型，自组织竞争型神经网络属于无监督的神经网络模型，一般包括三层，分别是输入层、竞争层以及输出层。“胜者为王，败者为寇”是自组织竞争型神经网络中神经元的激活或抑制的重要思想，竞争获胜的神经元的激活状态将会被加强，而竞争失败的神经元则保持不变。传统的自组织竞争型神经网络包括自组织映射（Self-Organizing Maps）网络、自适应谐振理论（Adaptive Resonance Theory）网络、对偶传播（Counter Propagation）网络等。

（1）自组织映射网络。自组织映射网络是一种典型的竞争型神经网络，常应用于非定向的数据挖掘任务，例如聚类等神经网络。1981 年由芬兰 Helsink 大学的 T • Kohonen 教授提出，因此自组织映射网络也常常被称作 Kohonen 网络。自组织映射网络提供了一种将高维数据在低维空间表示的方法。

（2）自适应谐振理论网络。自适应谐振理论模型是由美国 Boston 大学的 S・Grossberg 和 A・Carpenet 于 1976 年提出的网络模型。自适应谐振理论网络是无监督的学习网络。当神经网络和环境有交互作用时，对环境信息的编码会自发地在神经网中产生，可以理解为神经网络在进行自组织活动。

（3）对偶传播网络。对偶传播网络是美国著名学者 Robert Hecht-Nielsen 于 1987 年提出的。对偶传播网络能有效地存储二进制或模拟值的模式对，也可以用于联想以及模式分类等任务。

4.5 深度学习中常见的网络类型

4.5.1 卷积神经网络

卷积神经网络属于前面介绍的前馈神经网络之一，它对于图形图像的处理有着独特的效果，在结构上至少包括卷积层和池化层。卷积神经网络是最近几年不断发展的深度学习网络，并广泛被学术界重视和在企业中应用，代表性的卷积神经网络包括 LeNet-5、VGG、AlexNet 等。

目前卷积神经网络主要应用于影像中物体检测和识别、视频理解，除此之外，卷积神经网络还被应用于自然语言处理。实践证明，卷积神经网络可以有效地应用于自然语言处理中的语义分析、句子建模、分类等。同时，卷积神经网络还被应用于计算机围棋领域，在 2016 年 3 月 AlphaGo 对战李世乭的围棋比赛中，展示了包含卷积神经网络的深度学习在围棋领域的重大突破。

4.5.2 循环神经网络

不同于卷积神经网络，循环神经网络更擅长于对语言文本的处理。文本的分析处理，更看重时序上的输入与上下文的联系。循环神经网络的内部记忆结构，刚好满足这样的需求场景，因此在文本处理方面循环神经网络更胜一筹。此网络概念从提出到现在已经有二三十年历史，在理论与实践方面有不少积累。特别是 1997 年 LSTM 神经元的引入，解决了此网络模型的疑难问题，使得此网络在市场应用中广泛落地。

目前循环神经网络的主要落地场景在机器翻译、情感分析等 NLP 领域。特别是近几年媒体曝光较多的新闻写稿机器人，也是基于循环神经网络的一个应用。除此之外，循环神经网络还可以实现自动作诗、自动写歌词，甚至自动写代码。

4.5.3 深度信念网络

深度信念网络是于 2006 年由 Geoffreg Hinton 提出的神经网络结构，它是一种生成模型，

由多个受限玻尔兹曼机组成，采用逐层的方式进行训练，其结构可以理解为由多层简单学习模型组合而成的复合模型。深度信念网络是一个可以对训练的数据样本进行深层次表达的图形模型。

深度信念网络可以作为其他深度神经网络的预训练部分，主要做深度神经网络的权值初始化工作。众所周知，不合适的权值初始化方式会影响模型最终的性能，而采取预训练的方式可以尽可能更优地初始化权值，这样的方式可以有效提升模型的性能和收敛速度。

深度信念网络还可以衍生为其他类型的神经网络，例如卷积深度信念网络。卷积深度信念网络是目前深度学习中较新的发展分支，在结构上具有卷积神经网络的优势，在训练上也具备深度信念网络的预训练优势。

4.5.4　生成对抗网络

生成对抗网络将对抗的思想引入机器学习领域，对抗的双方为判别模型和生成模型。其中，判别模型的职责是准确区分真实数据和生成数据，而生成模型负责生成符合真实数据概率分布的新数据。通过判别模型和生成模型两个神经网络的对抗训练，生成对抗网络能够有效地生成符合真实数据分布的新数据。

生成对抗网络主要用于样本数据概率分布的建模，并生成与训练数据相同分布的新数据，例如，生成图像、语音、文字等。目前，GAN 主要应用于图像与视觉领域，以及自然语言处理领域，例如，提升图像分辨率、还原遮挡或破损图像、基于文本描述生成图像等。生成对抗网络为创造无监督学习模型提供了强有力的算法框架，未来将会更多地应用于无监督学习领域。

4.5.5　深度强化学习

深度强化学习是近几年深度学习中非常重要的技术领域，与其他机器学习的差异在于，深度强化学习更加注重基于环境的改变而调整自身的行为。Google 于 2015 年 2 月在 *Nature* 杂志上发表了“*Human-level control through deep reinforcement learning*”，详细阐述了通过深度强化学习计算机可以自己玩《Atari 2600》电子游戏。

深度强化学习的运行机制由四个基本组件组成：环境、代理、动作、反馈。通过四者的关系，强调代理如何在环境给予的奖励或者惩罚的刺激下，逐渐改变自己的行为动作，使得尽可能使用环境，从而达到环境给予的奖励值最大，逐步形成符合最大利益的惯性行为。

有监督学习在目前的工程实践中是比较成功的，但是对于处理一些难以学习和训练的特征，有监督学习并不能取得较好的效果。深度强化学习与传统的有监督学习方式不同，它不需要用户标记的数据作为训练集。深度强化学习更加注重在线规则，探索未知数据与环境之间的关系，并找到平衡点。

4.6 其他神经网络与深度学习

4.6.1 随机神经网络

随机神经网络（Stochastic Neural Networks）是由美国佛罗里达大学教授 Erol Gelenbe 于 1989 年提出的一种比较独特的网络结构，它在神经网络中引入了随机变化，在随机神经网络出现之前，众多神经网络属于确定性网络，表现在学习规则和运行过程，当输入确定时，结果也是确定的。神经网络中的随机性使得神经网络在单输入的条件下产生多输出的结果，不仅如此，随机噪声可以使得在梯度下降的过程中提供探索的可能性，为神经网络提供优化结果。

随机神经网络同样源于神经生物学的观点，按照神经生物学中的观点：生物神经元本质是存在随机性的。因为在神经网络中重复接受相同的刺激时，神经网络的响应并不相同，这意味着在生物神经网络中存在一定的随机性。

随机神经网络的随机性包括两方面：一方面是在神经元之间采用随机过程传递函数，另一方面是给神经元随机权值。由随机传递函数建立的随机神经网络常常被称作玻尔兹曼机。

4.6.2 量子神经网络

量子神经网络概念诞生于 20 世纪 90 年代后期，因为吸引了不同领域的科学家的关注，所以对于量子神经网络人们向不同方向进行了探索，提出了许多思路和初步模型，充分体现了量子神经网络研究的巨大潜力。主要研究方向可归纳为如下几个方面：

（1）量子神经网络使用神经网络的连接思想构建了量子计算机，通过神经网络模型研究量子计算中的问题。

（2）量子神经网络构建了基于量子计算机或量子器件的神经网络，通过使用超高速、超并行和数字级容量的特点，可以提高神经网络的结构和性能。

（3）量子神经网络作为传统计算机的混合智能优化算法，通过引入量子理论来改进传统神经网络的思想，通过使用量子理论的概念和方法，建立了新的网络模型，提高了传统神经网络的结构和性能。

（4）基于脑科学与认知科学的研究。

4.6.3 迁移学习

在大多数深度学习任务中，基本都会对数据进行训练，建立模型，最终将模型应用到各业

务场景中。但是往往会出现实际应用的效果比训练集和测试集上的效果差的情况，主要原因在于带人工标记的训练样本、测试样本都是有限的，无法囊括所有的数据特征。训练数据样本的分布不一定与实际业务中应用数据样本的分布一致，随着时间的变化，可能会根据实际业务中的情况，重新收集数据并重新构建新模型。

迁移学习（Transfer Learning）源于对数据的观测，期望通过以前学习到的知识应用于解决现有的新问题、适应新场景。迁移学习是近年来较为热门的技术之一，迁移学习的目的是将源领域学习到的模型应用到目标领域中去，源领域和目标领域存在一定的共性和差异性，简单而言，迁移学习将别处获得的经验模型，迁移到新的应用场景中，并能够很好地使用。

教育学中有一个概念叫作“学习迁移”，也就是说，如果一个学生学到了很多知识，那么验证其学习能力的方法就是看他是否有能力将学习的知识转移到新的场景中。一般而言，学习一个新的课程很容易，但转移学习能力到其他场景中是相对困难的。

迁移学习对于未来的机器学习方式有很大的影响，不仅会减少机器学习的学习成本，还会增强机器学习的通用性。例如，对于大陆和香港的开车方式，开车的技能是相通的，但是大陆的司机坐在汽车前进的左侧位置，而香港的司机坐在汽车前进的右侧位置，迁移学习则是将大陆学习的驾驶经验，应用于香港的交通驾驶实践中。

4.7　深度学习与多层神经网络的关系

在广义上，深度学习的网络结构也是一个多层神经网络。传统意义上的多层神经网络只是输入层、隐藏层、输出层。隐藏层中的层数取决于需要，截至目前，还没有明确的理论推导来解释到底有多少层是合适的。

深度学习一般而言，打破了传统意义上的多层神经网络，在传统多层神经网络的基础上，衍生出了新的层次，例如卷积神经网络中，在原有的多层神经网络基础之上，基于特征学习部分添加了卷积层和池化层。多层神经网络并不适用于所有场景，它有三个主要的限制：

（1）面对大量数据，需要手动提取原始数据的特征作为输入。必须忽略不相关的变量，同时保留有用的信息，在数据规模较大时难以把握。

（2）若想要更准确地逼近复杂的函数，则必须增加隐藏层的层数，这会导致梯度扩散的问题。

（3）无法处理时间序列数据（如音频），因为多层神经网络不包含序列的参数。

随着人工智能相关应用场景的不断增多，在更多领域，例如图像识别、视频理解、语义翻

译等，传统多层神经网络难以达到商业化效果。而深度神经网络模型在以下三个方面有一定优势，并可以辅助提升系统效果。

（1）深度学习的深度可以自动选择原始数据的特征。以图像识别为例，将像素值矩阵输入到深度卷积神经网络中，网络的第一层可以表示主要视觉信息，例如位置、边缘、对象的亮度等。网络的第二层可以将边缘整合，形成对象轮廓的特征，随着网络层次的加深，越深的层将表征更多的抽象信息。所有功能都自动在深度卷积神经网络中表达，而不是人为设计。更重要的是，随着网络层的深度不断加强，从具体到抽象的表征水平与大脑的工作原理一致，甚至超越人的抽象能力。

（2）深度学习训练算法的改进。一种方法是改变神经网络的组织结构。例如，在深度卷积神经网络中并不是完全采用全连接网络，训练算法仍然基于反向传播梯度的基本原理。另一种方法是完全改变训练算法，可以尝试各类优化算法，例如递归最小二乘法（RLS）等。

（3）灵活的神经网络结构。例如，使用具有反馈和时间参数的循环神经网络来处理时间序列相关的数据。在某种意义上，循环神经网络可以在时间维度上发展成深度神经网络，可以有效处理音频信息（例如语音识别）、自然语言处理等。

另外，初始值的选择和训练机制对深度学习的效果影响很大，传统神经网络采用反向传播方法，简单地使用迭代算法训练整个网络，可以随机设置初始值，然后根据当前神经网络的实际输出和期望输出之间的差异来改变先前层的参数，直到收敛（整体是梯度下降法）。

然而深度学习是一个层次训练的机制，如果使用反向传播机制，对于深层神经网络，随着深度的加深，反向传播的效果影响将越来越小，会有梯度扩散，从而影响精度。另外，应尽量避免随机初始化深度学习的模型参数，可以通过学习获得的输入数据的结构使得初始值更接近全局最优值，或者通过加载预训练的模型，也可以得到更好的结果。

4.8 调参技巧

深度学习中包含各种各样的参数，参数的调节对于模型的快速收敛以及达到最佳效果有着重要的作用，对于一些参数的调节方式，以下内容可以辅助参考。

（1）学习速率。对于大规模的数据训练，学习速率的默认值一般为较大值，例如 0.9。学习速率一般情况下不会太大，并且学习速率通常随着训练次数的增多而不断减小，衰减系数通常为 0.5。但是一般情况下，可以采用自适应梯度法，例如 Adam、AdaDelta、RMSprop 等。

（2）网络层次结构的设计。目前对于网络层次结构中的层数和神经元节点数并没有完整的

理论推断，但是建议从单层开始设计，每层神经元的节点数可以参照 16、32、64、128 这样的序列进行获取，一般情况下，某层神经元的节点数量不会太大。例如，某层神经元节点数超过 1000 的几乎没有。

（3）训练的批处理大小。批处理大小是训练过程中不可回避的问题，若批处理量设置过大，虽然可以提高模型的训练速度，但是模型的最终效果并不能得到保障；相反，若批处理量设置过小，虽然对最终结果的影响不会太大，但是会导致模型的收敛周期过长。一般情况下，可以设置批处理大小为 64 或 128。

（4）梯度裁剪阈值。梯度裁剪是在深度神经网络（例如循环神经网络）中防止梯度爆炸的有效方法，当参数矢量的 L2 范数超过一个特定阈值时可对参数矢量的梯度进行标准化，这个特定阈值是根据这个函数得来的：新梯度=梯度×阈值/梯度 L2 范数。

（5）正负样本采样比。正负样本的比例一般是比较容易忽视的，在分类模型中，理想情况下正负样本的比例为 1 比 1，但实际上，训练集并非理想化的训练数据，当发生正负样本采样比不为 1 比 1 时，模型的分类结果在很大程度上可能会倾向于训练集中占比较大的样本，从而影响模型的分类效果。可以对占比较小的样本数据加大采样比例，例如，对小数据进行复制以提高其比例。不仅对于整体样本是这样，对于模型训练的每一个批量也都应当尽可能让正负样本数量一致。例如，对于批处理大小为 64 的二分类训练数据，则批处理样本中正负样本数量应各为 32。上述方式不仅适用于二分类问题，对于多分类问题依然适用。

上面只是对调参过程中比较重要的参数进行了介绍，在实际工作中，会根据不同的数据采用不同的策略，甚至采用自动化调参、分阶段调参等方式对模型进行训练。

4.9　本章小结

本章全面介绍了神经网络的基础，从神经网络的学习方法入手，介绍了常见的学习规则，如误差修正学习、赫布学习规则、最小均方规则以及竞争学习规则。然后重点介绍了梯度下降优化方法，并对梯度下降优化方法的各种类型进行了介绍。接着对常见的神经网络类型，如前馈型神经网络、反馈型神经网络、自组织神经网络、随机神经网络等进行了介绍；同时，还对深度学习中的常见网络类型，如卷积神经网络、递归神经网络、循环神经网络等进行了概要介绍。本章最后对深度学习中的常见编码模型：深度自动编码、降噪自动编码等也做了相应介绍。

进阶篇

第 5 章

前馈型神经网络

前馈型神经网络是人工智能领域中最早发明使用的简单神经网络类型，也是目前应用最广泛、发展较为迅速的神经网络之一，从人工智能概念诞生之初，就被人研究，目前在学术理论和实践应用中都达到了较高的水平。

5.1 概述

前馈型神经网络作为一种简单的神经网络形式，各个神经元按照层次结构分层排列，层与层之间的神经元相互连接，前一层神经元的输出作为下一层神经元的输入，各层次之间没有反馈，层内之间的神经元没有任何连接，其中第一层被称作输入层，最后一层是输出层，中间其他层被称作隐藏层。整个神经网络结构无反馈，可以用一个有向无环图进行表示。无反馈并不是意味着神经网络中的参数不能反向传播，而是指神经网络的拓扑结构上不存在环路或者回路结构。

5.2 常见结构

常见的前馈型神经网络包括单层的前馈型神经网络结构以及多层的前馈型神经网络结构，单层的前馈型神经网络被称作单层感知器，多层的前馈型神经网络被称作多层感知器。

（1）单层感知器结构。单层感知器结构如图 5-1 所示，它仅由输入层和输出层组成，不包含隐藏层。

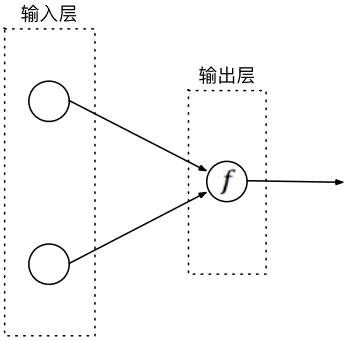


图 5-1　简单的单层感知器结构示例

（2）多层感知器结构。多层感知器结构在单层感知器结构的基础上，至少包含一个隐藏层，如图 5-2 所示。

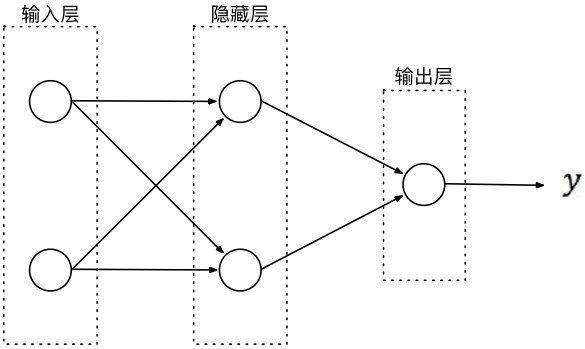


图 5-2　简单的多层感知器结构示例

5.3　单层感知器网络

5.3.1　原理

感知器（Perceptron）是科学家 Frank Rosenblatt 于 20 世纪 60 年代发明的，它是神经网络家族中最容易理解和应用的模型，也是最简单的前馈型神经网络，同时也是一种二元线性分类模型，主要用于模式分类。感知器在人工智能的第一次浪潮中比较流行，在当时的社会背景下，解决了很多现实而又实际的问题。

感知器是从生物学的神经网络中抽象出的算法模型。生物学中的神经细胞可以分为：树突、突触、细胞体及轴突，对于单个的神经网络细胞可以被视为仅有两种存在状态：一种是激活态，另种是抑制态。然而神经网络系统的状态取决于传递该神经元的信息量及突触。当信息量超越

某个阈值时，该神经细胞基处于激活态，同时神经元产生电脉冲，产生的电脉冲通过轴突和突触传递到其他的神经元细胞。感知器即模拟的生物学神经网络的过程，将突触视为权值，阈值视为偏置，将细胞体视为激活函数。

感知器实质是一种最简单形式的前馈型神经网络、单层神经网络，只要样本数据是线性可分的，则通过多次迭代，感知器算法就一定会实现收敛。在更多的应用场景中，很多样本是线性不可分的，直接采用感知器算法的场景相对较少，一般感知器算法会作为一种基础算法在复合算法中应用。

5.3.2 网络结构

感知器的网络结构如图 5-3 所示。

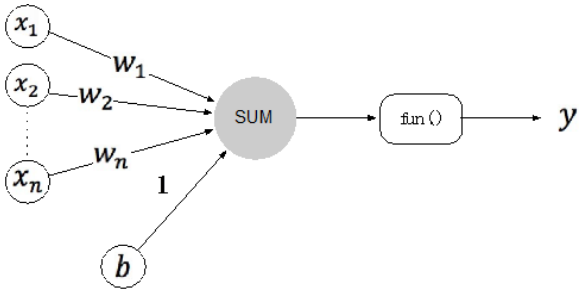


图 5-3 简单的感知器网络结构示例

图 5-3 表示一个n维的输入的单层感知器，x_1~x_n表示输入的n个向量，w_1~w_n对应每一个向量的权值，b为偏置，一个不依赖于任何输入向量的常量，fun(.)表示激活函数，偏置可以理解为激活函数的偏移量，最终y为输入n维向量之后的输出结果。

用公式表示如下：

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) = f(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X})$$

采用阶梯函数作为激活函数则模型结构为：

$$y = \begin{cases} +1, & \boldsymbol{W}^{\mathrm{T}}\boldsymbol{X} + b > 0 \\ -1, & \boldsymbol{W}^{\mathrm{T}}\boldsymbol{X} + b \leqslant 0 \end{cases}$$

5.3.3 实例一：基于单层感知器“与”运算

一个常见的例子就是利用感知器进行与运算，输入仅有的两个变量：x_1和x_2，进行与运算的结果如表 5-1 所示。

表 5-1　两个参数的与运算示例

x_1	x_2	y
1	0	0
0	0	0
1	1	1
0	1	0

根据公式$y=f(W^{\mathrm{T}}X)$，则很容易进行计算：$y=f(x_1\times w_1+x_2\times w_2+b)$，其中约定$w_1=0.5$、$w_2=0.5$、$b=-0.6$，激活函数采用如下公式：

$$f(x)=\begin{cases}1, & x\geqslant 0\\ 0, & x<0\end{cases}$$

则对于第一组输入$x=\{1,0\}$：$y=f(x_1\times w_1+x_2\times w_2+b)=f(1\times 0.5+0\times 0.5-0.6)=f(-0.1)=0$。

同理，对第三组输入$x=\{1,1\}$：$y=f(x_1\times w_1+x_2\times w_2+b)=f(1\times 0.5+1\times 0.5-0.6)=f(0.4)=1$。

因此，上述即是一个感知器对于数据运算的过程，同理，可以利用感知器进行或运算。但无论是利用感知器进行与运算还是或运算，都离不开确定每一个输入x_i对应的w_i，偏执项b可以理解为输入x_n的值为 1 的对应w_n，因此，如何计算w_n成为了感知器中比较关键的问题，需要借助感知器训练算法进行相关计算。

5.3.4　实例二：利用感知器判定零件是否合格

在某工厂中，有一组零件是否合格的数据，是否合格的关键在于零件的三个因素：长、宽、高。样品数据如表 5-2 所示。

表 5-2　零件的样本数据示例

长（单位：厘米）	宽（单位：厘米）	高（单位：厘米）	是否合格
2	2	1	合格
1	2	1	不合格
1	2	2	合格
1	1	2	不合格
……	……	……	……

激活函数采用如下公式：

$$f(x)=\begin{cases}1, & x\geqslant 0\\ 0, & x<0\end{cases}$$

可以将上述零件是否合格转换为 0、1 问题，合格约定为 1，不合格约定为 0。默认w_i均为 1、学习速率 0.1，下面基于单层感知器计算第一次迭代过程。

① 计算每一行根据默认的w_i计算激活函数需要的输入值z，感知器输出值$y=f(z)$，f为激活函数，如表 5-3 所示。

表 5-3 z值计算示例

样品编号	z值计算公式
1	z_1=2×w_1+2×w_2+1×w_3+b=2×1+2×1+1×1+1+1 = 6
2	z_2=1×w_1+2×w_2+1×w_3+b=1×1+2×1+1×1 +1= 5
3	z_3=1×w_1+2×w_2+2×w_3+b=1×1+2×1+1×1+1 = 5
4	z_4=1×w_1+1×w_2+2×w_3+b=1×1+1×1+2×1+1 = 5

② 计算目标期望输出t与样本感知器实际输出的差值$f(z)$的差值，如表 5-4 所示。

表 5-4 期望值与实际值之间的差值示例

样品编号	t	y=f (z)	t−y
1	1	1	0
2	0	1	-1
3	1	1	0
4	0	1	-1

③ 计算可能调整的Δw_i。根据公式$\Delta w_i=\eta(t-y)\times x_i$，分别计算四个样品的变化量，并相加变化值，结算过程如下所示：

Δw_1= 0.1×(0)×2+0.1×(-1)×1+0.1×(0)×1+0.1×(-1)×1=-0.2

Δw_2= 0.1×(0)×2+0.1×(-1)×2+0.1×(0)×2+0.1×(-1)×1=-0.3

Δw_3= 0.1×(0)×1+0.1×(-1)×1+0.1×(0)×2+0.1×(-1)×2 =-0.3

④ 计算本轮迭代后的 w1、w2、w3：

$$w_1=w_1+\Delta w_1=1+(-0.2)=0.8$$

$$w_2=w_2+\Delta w_2=1+(-0.3)=0.7$$

$$w_3=w_3+\Delta w_3=1+(-0.3)=0.7$$

通过上述过程完成了第一次的迭代，迭代后的w_i值已经发生变化，不断重复上述计算过程，

直到计算到达最大迭代次数或者误差小于误差阀值时即可停止迭代。这里的误差是指所有样品的$t-y$之和，即期望值与真实值的差距小到一定程度时，认为已经达到误差阀值。

5.4　BP 神经网络

5.4.1　概述

反向传播神经网络（Back Propagation Neural Network）也被称为多层感知器，是目前应用较为广泛的神经网络之一。

多层感知器相对于单层的感知器网络增加了隐藏层。隐藏层数量的选择目前还不具备理论支持。在多层感知器中，确定的是输入层和输出层的节点数量，隐藏层节点数量是不确定的，隐藏层节点的数量对神经网络的性能有影响，经验公式可以确定隐藏层节点的数量，如下：

$$h=\sqrt{m+n}+a$$

其中h是隐藏层节点的数量，m为输入节点个数，n是输出节点个数，a是 1～10 个常数之间的可调节常数。

对于无隐藏层的单层感知器，它的决策区域则为一个超平面划分的两个区域；而对于单隐藏层的感知器，它的决策区域则为一个开凸区域或者闭凸区域；对于多隐藏层的感知器，其决策区域相对比较任意，甚至可以模拟任意形状的划分。

5.4.2　反向传播算法

前馈型神经网络的训练事实上是不断调整网络中权值和偏置两类参数。前馈型神经网络的训练过程一般采用反向传播算法，反向传播算法可以分为正向传输和反向反馈两部分。正向传输负责逐层传输计算输出值，反向反馈则是根据输出值反向逐层调整网络的权值和偏置。对于一个如图 5-4 所示的多层感知器，它可以利用反向传播算法进行模型的训练。

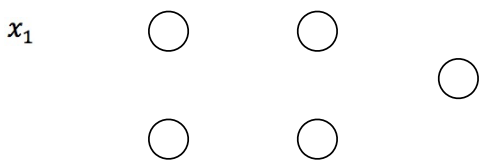


图 5-4　简单的多层感知器示例

其中x_1和x_2为输入层神经元，$f(z_1)$、$f(z_2)$、$f(z_3)$、$f(z_4)$为隐藏层神经元，$f(z_5)$为输出

层神经元。z_i表示神经元上层的输入，定义$y_i = f(z_i)$，表示神经元的输出值。

1. 正向传输

在训练网络之前，需要随机初始化权值和偏置，初始化权值为随机的[-1,1]区间实数，初始化的偏置为随机的[0,1]区间实数，然后开始向前传输。正向传输是从输入层x_1和x_2的神经元不断向输出层计算的过程。

对于$f(z_1)$的神经元，在不考虑偏置的情况下可以计算为：

$$y_1 = f(z_1) = f(w_{(x_1)1} \times x_1 + w_{(x_2)1} \times x_2)$$

$w_{(x_1)1}$表示x_1到y_1的权值，$w_{(x_2)1}$表示x_2到y_1的权值，如图 5-5 所示。

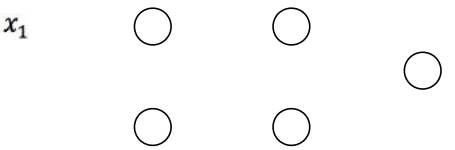


图 5-5　多层感知器中的权值参数示例

同理，可以计算：

$$y_2 = f(z_2) = f(w_{(x_1)2} \times x_1 + w_{(x_2)2} \times x_2)$$

$$y_3 = f(z_3) = f(w_{(y_1)1} \times y_1 + w_{(y_2)1} \times y_2)$$

$$y_4 = f(z_4) = f(w_{(y_1)2} \times y_1 + w_{(y_2)2} \times y_2)$$

$$y_5 = f(z_5) = f(w_{(y_3)1} \times y_3 + w_{(y_4)1} \times y_4)$$

通过正向传播，可以计算出每一个神经元节点的输出值，最终的输出值y_5即为当前正向传播模型的实际输出。

2. 反向反馈

反向反馈则是从神经网络的输出层，向前推导调整权值的过程。在正向传输过程中，已经获得模型的实际输出值y_5，假定训练数据的期望输出值为t，则实际输出值和期望输出值之间存在误差δ，定义$\delta = t - y_5$，误差定义的方式可以根据实际情况定义。假定每一个神经元均存在误差，且定义每一个神经元的误差为δ_i，例如y_4神经元的误差为δ_4。在反向传播过程中很重要的两个任务即为计算误差和调整权值，对于误差的表示，如图 5-6 所示。

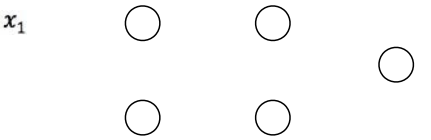


图 5-6　神经网络中的误差表示示例

在反向传播过程中计算δ_i时，权值复用正向传输的权值，对于误差$\delta_3 = w_{(y_3)}\delta$，同理$\delta_4 = w_{(y_4)}\delta$。而对于$\delta_1$和$\delta_2$的计算方式如下：

$$\delta_1 = w_{(y_1)1}\delta_3 + w_{(y_1)2}\delta_4$$

$$\delta_2 = w_{(y_2)1}\delta_3 + w_{(y_2)2}\delta_4$$

通过上述计算，即可获得δ_1、δ_2、δ_3、δ_4的值。反向传播的最终目的还是计算神经元之间的权值变化，误差是调整权值中非常重要的参数，对于权值调整的变化量可以定义为：

$$\Delta w_i = \eta\delta_i \frac{\mathrm{d}f(z_i)}{\mathrm{d}z_i} x_i$$

其中η为学习速率，因此对于$w_{(x_1)1}$权值的调整，可以参照如下公式：

$$w'_{(x_1)1} = w_{(x_1)1} + \eta\delta_1 \frac{\mathrm{d}f(z_1)}{\mathrm{d}z_1} x_1$$

同理，对于$w_{(x_2)1}$权值的调整为：

$$w'_{(x_2)1} = w_{(x_2)1} + \eta\delta_1 \frac{\mathrm{d}f(z_1)}{\mathrm{d}z_1} x_2$$

每一个权值按照上述公式，即可完成对权值的更新，神经元之间的权值更新完毕之后，则反向传播算法结束。反向传播算法的过程实际是完成样本对模型的调参，不断地重复正向传输和反向反馈过程，误差会越来越小，权值也会越来越稳定，模型的准确率也会越来越高。

上述即为神经网络每个节点误差的计算和权值更新的方法。从中可以看出计算一个节点的误差值，首先需要计算每个神经元与其相连的下一层节点的误差值，因此误差值的计算顺序必须是从输出层开始计算，然后反向依次计算每个隐藏层神经元的误差值，直到第一个隐藏层，反向传播算法的含义即为如此。

随着神经网络的发展，更新权值的方式已经比较简单和成熟，归纳起来就是求梯度和梯度下降，梯度的方向指明了误差扩大的方向，因此在更新权值时需要对其取反，从而减少权值导致的误差。在实际工程应用中，权值的更新方式也会根据实际情况调整权值更新策略。

倘若一个神经网络是一个分类模型，则最后的输出层应当可以描述数据记录的类型，例如，

对于二分类问题，可以使用一个神经元作为输出层，如果输出层的神经元的输出概率值大于阈值，则可以认为该数据记录属于某个类别，反之则不属于该类别。反向传播算法不仅可以用于前馈型神经网络，还可以用于其他神经网络的训练。

3. 终止条件

通过正向传输和反向反馈，已经完成了训练神经网络的过程，然后不断训练模型并完善，但训练的过程不是无休止的，而是有停止条件的。目前有两种类型的停止条件：① 设置最大迭代次数，如训练使用数据集迭代 100 次后停止；② 计算训练集对网络预测精度，在训练达到一定阈值后主动停止。

反向传播算法是感知器训练的重要方法，一般将采用反向传播算法进行训练的多层感知器称作“BP 神经网络”，即 BP 神经网络是一种基于反向传播算法多层前馈网络。

5.4.3 异或问题的解决

当激活函数选择阶跃函数时，激活函数表示为sgn，它的输出判定规则为：当 x>0 时，最终结果输出 1，反之则输出 0，设定偏置值分别为-1.5、-0.5、-0.5。神经元与神经元之间的权值如图 5-7 所示。

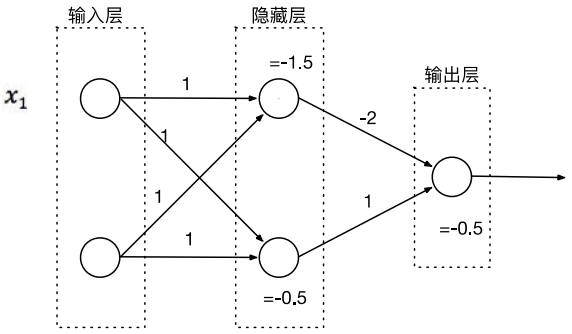


图 5-7 异或问题的解决示例

因此对输入的x_1和x_2的情况，当$x_1 = 1$，$x_2 = 1$时，有：

$$f_1 = \text{sgn}(x_1 \times 1 + x_2 \times 1 + b_1)$$

$$= \text{sgn}(1 \times 1 + 1 \times 1 - 1.5)$$

$$= \text{sgn}(0.5)$$

$$= 1$$

$$f_2 = \text{sgn}(x_1 \times 1 + x_2 \times 1 + b_2)$$
$$= \text{sgn}(1 \times 1 + 1 \times 1 - 0.5)$$
$$= \text{sgn}(1.5)$$
$$= 1$$
$$f = \text{sgn}(f_1 \times -2 + f_2 \times 1 + b_3)$$
$$= \text{sgn}(1 \times -2 + 1 \times 1 - 0.5)$$
$$= \text{sgn}(-1.5)$$
$$= 0$$

同理，当$x_1 = 1$，$x_2 = 0$时，有：

$$f_1 = \text{sgn}(x_1 \times 1 + x_2 \times 1 + b_1)$$
$$= \text{sgn}(1 \times 1 + 0 \times 1 - 1.5)$$
$$= \text{sgn}(-0.5)$$
$$= 0$$
$$f_2 = \text{sgn}(x_1 \times 1 + x_2 \times 1 + b_2)$$
$$= \text{sgn}(1 \times 1 + 0 \times 1 - 0.5)$$
$$= \text{sgn}(0.5)$$
$$= 1$$
$$f = \text{sgn}(f_1 \times -2 + f_2 \times 1 + b_3)$$
$$= \text{sgn}(0 \times -2 + 1 \times 1 - 0.5)$$
$$= \text{sgn}(0.5)$$
$$= 1$$

同理，对x_1和x_2的其他情况也按照上述方式进行计算，因此，最终得到的异或值如表 5-5 所示。

表 5-5　两个变量的异或值对应表

输入x_1	输入x_2	异或结果
1	0	1
1	1	0
0	0	0
0	1	1

按照异或运算符，即为：0⊕0=0、0⊕1=1、1⊕0=1、1⊕1=0，这样的异或运算也可以理解为异或问题的分类，如图 5-8 所示。

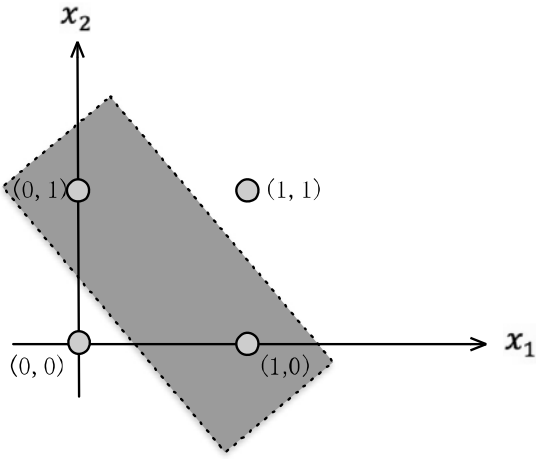


图 5-8　异或问题的分类

5.4.4　避免病态结果

一般情况下，神经网络能够较好地训练样本数据，相反的情况则是神经网络无法有效地训练样本数据，这通常是由神经网络的病态问题（ill-conditioning）导致的。在多层感知器的训练过程中，神经网络的病态问题一般与训练样本数据、神经网络结构以及神经网络的初始权值有一定关系。常见的原因是输入训练样本数据过大、神经网络层结构大小不一、初始权值赋值不合理（过大或过小）。

病态问题结果是计算的结果通用性或兼容性不够强，很容易因参数的变化导致结果发生较大变化。

以病态方程组为例，病态方程组是指因系数的很小改变却导致解改变很大的方程组，称相应的系数矩阵为病态矩阵。由于根据实际问题建立的方程组的系数矩阵或者常数向量的元素本身会存在一定的误差，而这些初始数据的误差在计算过程中就会向前传输，从而影响方程组的

解。病态方程组对任何算法都将产生数值不稳定性。对病态方程组可以采用：高精度的算术运算、预处理的方式以及采用特殊的数值解法来解决病态问题。

例如对于如下两组方程：

$$\begin{cases} x+y=2 \\ x+1.001y=2 \end{cases} \qquad \begin{cases} x+y=2 \\ x+1.001y=2.001 \end{cases}$$

从方程的形式上看，两组方程组的第一个方程相同，不同的是第二个方程的右边值，分别为 2 和 2.001。两组方程本身在系数上并没有较大的改动，但是通过计算，左边的方程组的解为$x=2$、$y=0$，而右边的解为$x=1$、$y=1$。两组方程本身因为系数的极小变化，导致了解的完全不同且相差较大。

对于另外三组方程组：

$$\begin{cases} x+y=2 \\ 1.001x+y=1 \end{cases} \quad \begin{cases} x+y=2 \\ 0.999y=0.998 \end{cases} \quad \begin{cases} x+y=2 \\ y=1.00 \end{cases}$$

对上面三组方程组进行求解，可以发现三组方程得到相对准确的$x=1$以及$y=1$的解，方程组存在一定的差异，求得的解的差异也较小，这样的方程会使得系统相对稳定。

上述类似的病态问题应当尽量避免，否则在神经网络的训练过程中，迭代训练的效果会受到较大的影响，甚至在模型预测过程中也会导致很大的差异，一个可靠稳定的神经网络模型对于商业化应用非常重要。此外，病态问题可以通过牛顿法、最速下降法等方式进行优化解决。

5.4.5　实例：基于多层感知器的手写体数字识别

假定需要判断一张手写数字图像上面的数字，因此需要将图像作为多层感知器的输入。可以将图像像素的灰度值作为多层感知器的输入。以手写体训练集 MNIST 为例，图像是 28×28 的灰度图像，因此需要 784 个输入神经元，而每个神经元的值为 0 到 1 间的图像灰度值。输出层包含 10 个神经元，每个神经元的输出值表示手写体图像为当前神经元对应数字的概率值，10 个神经元的输出之和为 1，网络设计结构如图 5-9 所示。

x_2

图 5-9 基于多层感知器手写体识别网络结构示例

在如图 5-9 所示的神经网络结构中，设定L表示神经网络的层数；n^l表示第l层神经元的数量；$\sigma_l(\cdot)$表示第l层神经元的激活函数；$W^l \in \mathbb{R}^{n^{l-1} \times n^l}$表示第$l-1$层到第$l$层神经元的权值矩阵；$b^l \in \mathbb{R}^{n^l}$表示第$l$层神经元的偏置量；$z^l \in \mathbb{R}^{n^l}$表示第$l$层神经元的输入；$a^l \in \mathbb{R}^{n^l}$表示第$l$层神经元的输出。

多层感知器的神经网络由输入层和输出层组成，因此$L = 2$。

（1）输入层：由 784 个神经元组成，$n^1 = 784$。将 28×28 的手写体图像按像素展开为一维列向量$\boldsymbol{z^1} = \begin{bmatrix} \text{img}_1 \\ \vdots \\ \text{img}_{784} \end{bmatrix}$，将展开后的列向量$\boldsymbol{z^1}$传入输入层。

（2）第二层，输出层：由 10 个神经元组成，$n^2 = 10$。输入层和输出层之间采用全连接，用$w_{i,j}^2$表示输入层第i个神经元到输出层第j个神经元的权值，用b_j^2表示输出层第j个神经元的偏置量，因此$\boldsymbol{z^2} = [\boldsymbol{W}^2]^{\mathrm{T}} \cdot a^1 + b^2 = [\boldsymbol{W}^2]^{\mathrm{T}} \cdot z^1 + b^2$，其中$\boldsymbol{W}^2 = \begin{bmatrix} \boldsymbol{w}_{1,1}^2 & \cdots & \boldsymbol{w}_{0,10}^2 \\ \vdots & \ddots & \vdots \\ \boldsymbol{w}_{784,1}^2 & \cdots & \boldsymbol{w}_{784,10}^2 \end{bmatrix}$，$\boldsymbol{b^2} = \begin{bmatrix} b_1^2 \\ \vdots \\ b_{10}^2 \end{bmatrix}$。输出层采用 softmax 作为激活函数，因此$a^2 = \sigma_2(z^2) = \text{softmax}(z^2)$。

定义损失函数为期望类别和预测类别之间的交叉熵，公式如下，其中y为期望类别：

$$\text{loss} = -\sum_{i}^{n} [y \cdot \log(a^L)]$$

以 50 个样本为一组进行训练，共迭代训练 1000 次，反向传播的优化函数选用最速下降法，学习速率设置为 0.01。最后，用测试集数据来验证训练好的模型，实践表明，基于多层感知器的手写体数字分类准确率约为 91%。

5.5　径向基函数神经网络

5.5.1　原理介绍

径向基函数（Radical Basis Function，RBF）神经网络是一种以径向基函数为激活函数的神经网络，它具有近似模拟能力强、分类能力强和学习速度快等优势。1985 年，Powell 提出了多变量插值的径向基函数方法。1988 年，Moody 和 Darken 提出了一种神经网络结构，即径向基函数神经网络，它能够以任意精度逼近任意连续函数，特别适合解决分类问题。

径向基函数其实就是沿径向对称的标量函数，一般情况下定义为空间任意点x到中心c之间的欧氏距离的单调函数，可以记作$k(||x-c||)$。最常用的径向基函数是高斯核函数，对于给定了两个样本x和x'的径向基函数和可以表示为某个输入空间的特征向量，可以定义为如下公式：

$$k(||x-x'||)=\exp(-\frac{||x-x'||^2}{2\sigma^2})$$

其中$||x-x'||^2$可以表示为两个特征向量之间的平方欧几里得距离，σ是一个自由参数，可以表示为函数的宽度参数，控制函数的径向作用范围，若定义$\gamma=-\frac{1}{2\sigma^2}$，则上述公式可以简化为如下公式：

$$k(||x-x'||)=\exp(\gamma||x-x'||^2)$$

根据上述公式，可以看出核函数的值随着距离的增加而减小，并介于(0,1)区间，因此可以理解为径向基函数是一种相似性的度量表示方法。常见的径向基函数包括高斯函数（Gaussian Function）、多二次函数（Multiquadratic Function）和逆二次函数（Reciprocal Multiquadratic Function）。

（1）高斯函数公式如下：

$$\varphi_i(x)=\exp(-\frac{||x-c_i||^2}{2\sigma^2})$$

（2）多二次函数公式如下：

$$\varphi_i(x)=(||x-c_i||^2+\sigma^2)^{-\frac{1}{2}}$$

（3）逆二次函数公式如下：

$$\varphi_i(x)=||x-c_i||^2\log(||x-c_i||)$$

一个径向基函数神经网络的结构如图 5-10 所示。

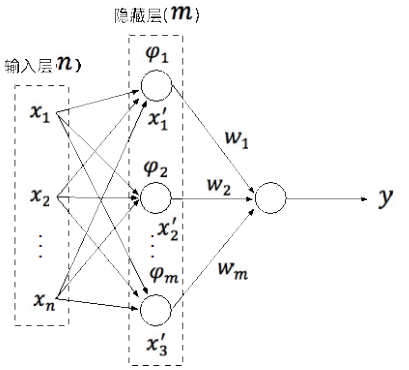


图 5-10　一个简单的径向基函数神经网络结构

径向基函数神经网络的结构，与具备单隐藏层的感知器结构并无太大差异。径向基函数神经网络是一个具有单个隐藏层的三层前向网络。第一层是由信号源节点组成的输入层。第二层是隐藏层，隐藏层节点数量取决于所描述问题的需要。隐藏层神经元的激活函数，即径向基函数。具有径向对称性和中心点衰减的非负线性函数，是局部响应函数，特定局部响应反映在可见层向隐藏层，不同于其他网络。以前的前向网络变换功能是全局响应的功能。第三层是输出层，输出层是对线性权进行调整，采用线性优化策略，学习速度比较快，然而隐藏层调整激活函数的参数采用的是非线性优化策略，从而学习速度慢，并且它的参数初始化有一定的方法，而不是随机初始化。

在图 5-10 中，输入层的X为n维的向量，隐藏层为m维向量，φ_i表示隐藏层中第i个节点的径向基函数，x_i'表示第i个节点的径向基函数的中心点，且x_i'来自于输入层。单隐藏层的作用可以理解为对输入层向量的映射，当$m > n$时，表示将输入层的低维数据映射到高维空间，这样做的优势在于，在低维度中不可划分的数据，在高维数据空间中可能可以实现划分，这也是径向基函数神经网络的基本思想。

从图 5-10 以及公式定义中可以看出，径向基函数神经网络在学习训练过程中，包含三个重要参数的训练，分别是径向基函数的中心、宽度参数σ和隐藏层对输出层的权值。

5.5.2　中心选择方法

径向基函数神经网络的学习算法的关键问题是隐藏层神经元中心参数的合理确定。常用的方法是从给定的训练样本集中直接选择中心参数（或其初始值），或通过聚类方法进行选择，一般有以下几种方法辅助进行中心选择。

（1）直接计算方式。随机选择径向基函数中心，在输入样本中随机选择隐藏层神经元的中心，确定中心。一旦中心被确定，隐藏层中的神经元的输出便是已知的，并且这样的神经网络

的连接权值可以通过求解线性方程来确定。

（2）自组织学习方式。是否为中心由自组织学习决定。输出层的线性权值由监督学习确定。这种方法是神经网络资源的重新分配，通过学习，使得径向基函数神经网络隐藏层神经元的中心在输入空间的重要领域。该方式主要采用 K 均值聚类方法选择径向基函数神经网络中心，属于无监督的学习方法。

（3）有监督学习通过训练样本集选择径向基函数中心，以获得满足监督要求的网络中心和其他权值参数，常用的方法是梯度下降法。

此外，正交最小二乘法（Orthogoal Least Square）也可以用于中心选择。值得说明的是，在实际应用过程中，径向基神经网络隐藏层神经元的中心并不一定是训练样本中的样本集的聚类中心，而是通过学习训练得到的中心，这样的中心有助于更好地反映出训练样本集的各类特征和信息。

5.5.3　训练过程

对于径向基神经网络的训练过程则主要分为两个部分，第一步确定隐藏层的神经元数量、确定中心以及宽度参数；第二步则是计算隐藏层神经元与输出层神经元之间的权值。

径向基函数神经网络的训练方式多种多样，第一步可以通过无监督的方式得到确定的神经元数量和中心点、宽度参数；第二步则可以通过有监督的学习方式获得各项权值。按照无监督学习和监督学习混合的方式，训练过程大致如下。

1. 确定输入向量$\boldsymbol{X}=[x_1,x_2,\ldots,x_n]^{\mathrm{T}}$，其中$n$表示输入层的单元数量；确定期望的输出$\boldsymbol{O}=[o,o_2,\ldots,o_p]^{\mathrm{T}}$，其中$p$表示输出层数量。

2. 在输入向量$\boldsymbol{X}$即训练样本中随机选择h个样本分别作为h个径向基函数的中心，并通过聚类算法，例如通过 K-Means 或 DBSCAN 等聚类方式得到h个聚类中心，将这些聚类中心当作径向基函数的h个中心。

3. 计算宽度参数，宽度参数实际是方差。设定σ_i为第i个径向基函数的方差，假定径向基函数神经网络的基函数采用的是高斯函数，则方差的公式则为：$\sigma_i=\frac{c_{\max}}{\sqrt{2h}}(i\in[1,h])$。其中$c_{\max}$表示$h$个聚类中心之间的最大距离。

4. 隐藏层与输出层神经元之间的链接权值，则可以利用最小均方误差计算得到。

上述过程采用的是监督学习和非监督学习混合的方式，也可以采用纯监督学习的方式对神经网络进行训练，训练参数依然包括三部分：径向基函数的中心、宽度参数（方差）以及隐藏层神经元与输出层神经元之间的权值。采用监督学习的方式时，主要是对代价函数进行梯度下

降，然后逐一调整参数值，训练过程大致如下。

① 与上一种方式一样，确定输入向量$\boldsymbol{X}=[x_1,x_2,\ldots,x_n]^{\mathrm{T}}$以及期望的输出$\boldsymbol{O}=[o,o_2,\ldots,o_p]^{\mathrm{T}}$。

② 随机选择径向基函数的中心、方差以及隐藏层与输出层之间的权值。对于径向基函数的中心也可以采用上一种方式进行初始化。隐藏层与输出层的权值初始化方法也可以采用中心初始化的方式。

③ 通过梯度下降的方式对神经网络中的三类参数进行调整优化，代价函数采用均方误差的形式，计算的是神经网络的实际输出与期望输出的均方误差，公式为$e_i=\frac{(y(x_i)-o_i)^2}{2}$，$y(x)$为神经网络的实际输出，$o$为神经网络的期望输出。

④ 计算梯度并在梯度的反方向基础上调整三类参数，得到的公式分别如下：

$$\Delta c_k=-\eta\frac{\partial e}{\partial c_k}=\eta\frac{w_k}{\sigma_k^2}\sum_{i=1}^{p}e_i\varphi(||x_i-c_k||)(x_i-c_k)$$

$$\Delta\sigma_k=-\eta\frac{\partial e}{\partial\sigma_k}=\eta\frac{w_k}{\sigma_k^3}\sum_{i=1}^{p}e_i\varphi(||x_i-c_k||)||x_i-c_k||^2$$

$$\Delta w_k=-\eta\frac{\partial e}{\partial w_k}=\eta\sum_{i=1}^{p}e_i\varphi(||x_i-c_k||)$$

上述公式中φ表示基函数，可以选择高斯函数或其他函数，η为学习速率。通过上述过程的不断迭代即可完成径向基函数神经网络的三类参数计算。

5.5.4 径向基函数神经网络与 BP 神经网络的差异

在神经网络中参数调整时，存在全局逼近和局部逼近两种方式。全局逼近表示需要对网络的所有参数进行修正，实际原因是网络中一个或者多个可调参数对其他输出存在影响，由于每一个参数的调整都会影响到其他参数，因此全局逼近的训练速度相对较慢。而局部逼近只需要对局部参数进行修正，训练速度相对较快。径向基函数神经网络属于局部逼近，而 BP 神经网络则属于典型的全局逼近。

BP 神经网络的隐藏层节点采用输入向量和权值向量的内积作为激活函数的独立变量，激活函数一般采用 Sigmoid 函数，而隐藏层节点对 BP 网络的输出效果相同，所以 BP 神经网络是非线性映射的全局逼近。 径向基函数神经网络的隐藏层节点使用输入向量和中心向量之间的距离作为激活函数的独立变量，并使用径向基函数作为激活函数，所以径向基函数神经网络属于局部逼近。

径向基函数神经网络和 BP 神经网络均为非线性多层前馈网络，它们属于一般的函数逼近

器。对于任何 BP 神经网络，总是有一个径向基函数神经网络可以替代它，反之亦然。但是两个网络在网络结构、训练算法、网络资源利用和性能等方面存在一定的差异。

（1）网络结构方面：首先，BP 神经网络的输入层与第一个隐藏层之间是基于权值的连接；而径向基函数神经网络采用的则是输入层与隐藏层的直接连接方式，隐藏层到输出层才实现基于权值的连接。其次，BP 神经网络的神经元激活函数一般采用非线性函数，而径向基函数神经网络的神经元激活函数一般采用径向基函数。最后，BP 神经网络一般是一个三层或三层以上的前馈型神经网络，且 BP 神经网络中的隐藏层层数和隐藏层的神经元数量并不确定，没有确定的规律和方式可寻，当确定了隐藏层层数和神经元数量后，在训练过程中则不再改变；而径向基函数神经网络是确定的三层前馈型神经网络，仅存在一层隐藏层，隐藏层的神经元数量会根据实际情况进行调整。

（2）训练算法方面：BP 神经网络采用反向传播算法对模型进行训练，但是反向传播算法存在一些不足，例如，容易陷入局部最小值、学习过程收敛速度慢、隐藏层和隐藏层神经元的数量难以确定等。BP 神经网络模型的训练效果对于网络结构非常敏感，而径向基函数神经网络可以动态地确定网络结构，支持在线和离线的训练方式，训练速度快，训练性能上优于 BP 神经网络。

（3）网络资源方面：从径向基函数神经网络的原理、结构以及训练算法可以看出，其隐藏层神经元的数量由训练数据的样本量、样本类别以及分布情况决定。例如，当采用最近邻集群的方法来训练网络模型时，网络中隐藏层神经元的分配仅与训练样本的分布以及隐藏层神经元的宽度有关，而不受具体执行任务的影响。

总体而言，径向基函数神经网络可以根据具体问题确定相应的网络拓扑，具有自学习、自组织以及自适应能力。它对非线性连续函数具有一致的逼近性，学习速度快，可以进行广泛的数据融合，可以高速地处理数据。径向基函数神经网络的特性使其具有比 BP 神经网络更强的活力。目前，径向基函数神经网络已经成功应用于非线性函数逼近、时间序列分析、数据分类、模式识别等，这种新形式的前馈型神经网络，具有最佳逼近以及全局最优的效果，同时可以快速迭代训练且不存在局部最优问题，这使得径向基函数神经网络在非线性时间序列的预测中得到了广泛的应用。

5.6　本章小结

本章从前馈型神经网络的基本内容入手，逐步介绍了常见的前馈型神经网络结构，并重点介绍了单层感知器网络、BP 神经网络以及径向基函数神经网络。单层感知器网络是整个前馈

型神经网络中最为基础的网络结构，是理解后续内容的基础；BP 神经网络是最为常见的前馈型神经网络，也被称作多层感知器，在实际业务中应用广泛，也是后续神经网络发展和演变的基础，并深入介绍了 BP 神经网络中经典的反向传播算法。径向基函数神经网络是另一种新颖的前馈型神经网络，其结构特殊性在于不仅只有一层隐藏层，还将输入向量直接映射到隐藏层，并且隐藏层到输出层之间采用线性映射。

前馈型神经网络是人工智能领域中最为简单的神经网络类型，是神经网络的重要基础。

第 6 章

反馈型神经网络

反馈型神经网络（Feedback Neural Network）也被称作自联想记忆网络、回归网络，是一种反馈动力学系统。在这种网络中，每个神经元同时将自身的输出信号作为输入信号反馈给其他神经元，并且通过工作一段时间使得系统达到稳定的状态。

6.1 概述

6.1.1 基本原理

反馈型神经网络是一种从输出到输入具有反馈连接调节的神经网络，其结构比前馈型神经网络更加复杂。

前馈型神经网络具有较强的学习模拟能力，结构简单易于实现，但是前馈型神经网络缺乏反馈。联想特性是神经网络的重要特性，主要包括联想映射能力和联想记忆能力，前馈型神经网络具有强大的联想映射能力，但是缺乏联想记忆能力，而反馈型神经网络具备联想记忆和优化计算的能力。

在反馈型神经网络中，输入的信号信息决定了整个反馈系统的初始状态，神经网络系统通过一系列的状态转换之后，逐渐达到一个平衡状态，这个平衡状态即为神经网络计算后输出的结果。状态的稳定性是反馈型神经网络中最重要的问题之一。

一个简单的反馈型神经网络如图 6-1 所示，与前面介绍的前馈型神经网络在结构上的不同点在于最后一层对第一层存在反馈，如果将反馈路径与正向的传输进行贯通，则反馈型神经网络是一个带环向或循环的神经网络结构。

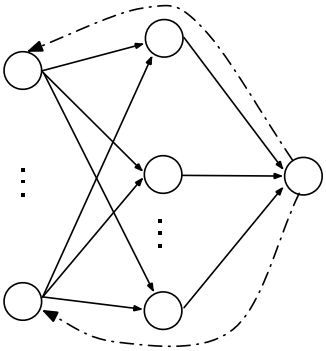


图 6-1 简单的反馈型神经网络

前馈型神经网络和反馈型神经网络均是确定性的网络结构，这种确定性表现于当确定输入层的输入后，则输出层的输出一定是确定性的。

反馈神经网络模型可以用无向图来表示。从系统的角度而言，反馈型神经网络是一种反馈动力学系统，它具有极复杂的动力学特性。在反馈型神经网络模型中，稳定性是神经网络联想记忆能力性质的体现；从计算的角度而言，反馈型神经网络模型具有比前馈型神经网络模型更强的计算能力，典型的反馈型神经网络包括 Hopfield 神经网络、海明神经网络、递归神经网络、双向联想存储器、Elman 神经网络。

6.1.2 与前馈型神经网络的差异

反馈型神经网络不同于前面介绍的前馈型神经网络。

首先，在输入变量方面，前馈型神经网络采用连续或者离散变量，一般不考虑输出与输入在时间上的滞后效应，只表达输出与输入的映射关系；反馈型神经网络不仅可以用离散的数据变量，还可以使用连续取值，考虑输出与输入之间在时间上的延迟，需要用动态方程来描述系统的模型，它是一种非线性动力学系统。

其次，在学习规则方面，前馈型神经网络的学习主要采用误差修正法的方式进行调参，例如，通过反向传播算法，计算迭代周期相对较长，导致收敛速度也比较慢；而反馈型神经网络主要采用赫布学习规则，一般情况下，计算的收敛速度比较快。

最后，在训练终止条件方面，前馈型神经网络通过达到一定迭代次数或者训练收敛从而结束训练过程；而对于反馈型神经网络，则是通过能量函数来判定模型是否趋于稳定状态。

反馈型神经网络与前馈型神经网络在应用上有一定的相似性，但在联想记忆和优化计算方面，反馈型神经网络更具优势。前馈型神经网络适用于联想映射和分类，而反馈型神经网络用于联想记忆和约束优化问题的求解。

6.2　Hopfield 神经网络

Hopfield 神经网络是 1982 年由约翰·霍普菲尔德提出的一种单层的反馈型神经网络模型，分为离散型 Hopfield 神经网络（Discrete Hopfield Neural Network，DHNN）和连续型 Hopfield 神经网络（Continuous Hopfield Neural Network，CHNN）。霍普菲尔德最早提出的 Hopfiled 神经网络是二值神经网络，神经元的输出值只取 1 或–1，分别表示神经元的激活或抑制状态。

离散型的 Hopfiled 神经网络的结构是一个全连接且不存在自环的无向图，是一种单层、输出为二值的反馈型神经网络。这种单层的网络由n个神经元组成，并且每一个神经元既是输入单元，又是输出单元。

一个简单的离散型 Hopfield 神经网络如图 6-2 所示。

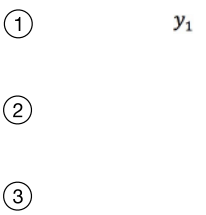


图 6-2　简单的离散型 Hopfield 神经网络

离散型Hopfiled神经网络仅存在单层三个神经元的简单结构，图中的①、②、③均表示同层的神经元，x_1、x_2、x_3是输入参数，y_1、y_2、y_3是输出参数，输出参数会进行反馈调节，设定y_i对①、②、③神经元的影响权值分别为w_{i1}、w_{i2}、w_{i3}，缩写为w_{ij}，其中i为输出值编号，j为神经元编号。因此对于神经元，它的计算公式如下：

$$\mu_j = \sum_i w_{ij} y_i + x_j$$

考虑到神经元本身的输出y_i是二值表示，因此可以通过阈值的方式进行调整，可以定义阈值函数f，当神经元的输出不小于阈值θ时，则输出为 1；当神经元的输出小于阈值θ时，则为–1。阈值函数f可以定义为如下：

$$\begin{cases} y_j = 1, \mu_j \geqslant \theta \\ y_j = -1, \mu_j < \theta \end{cases}$$

在离散型网络中，每个神经元都有相同功能，其输出称为状态。一个输出层为n的离散型

Hopfiled 神经网络状态的输出是神经元信息的集合，在t时刻的状态为一个n维的向量：

$$\boldsymbol{Y}(t) = [y_1(t), y_2(t), y_3(t), \dots, y_n(t)]^{\mathrm{T}}$$

由于神经元的输出仅可取值 1 或-1，因此对于t时刻的输出$\mathrm{Y}(t)$则存在2^n种状态信息。设定$y_j(t)$表示在t时刻节点j的状态，则在$t+1$时刻节点j的状态可以通过如下公式进行求解：

$$y_j(t+1) = f(\mu_j(t))$$

$$\mu_j(t) = \sum_{i=0}^{n} w_{ij} y_j(t) + x_j - \theta_j$$

上述公式中f依然为阈值函数，θ为阈值向量，$\theta = [\theta_1, \theta_2, \theta_3, \dots, \theta_n]^{\mathrm{T}}$。通过上述公式将时间序列$t$引入可以看出，离散型 Hopfiled 神经网络具有一定的记忆功能，当初始态确定后，离散型 Hopfiled 神经网络的状态按照工作规则向能量递减的方向逐渐变化，通过不断迭代训练使得系统接近或达到平衡点。如果$\Delta t > 0$，当网络从$t = 0$开始有初始状态$Y(0)$，当经过有限的时刻t之后，存在：$Y(t+\Delta t) = Y(t)$，则称网络是稳定状态。

在前面提到的公式中，也隐藏着两个重要的特征。一方面如果神经元和自身不存在相连，则w_{ii}恒等于 0，此时的离散型 Hopfiled 神经网络为无自反馈网络，相反为自反馈网络；另一方面连接权值是对称的，即$w_{ij} = w_{ji}$。权值的对称是离散型 Hopfiled 神经网络的重要特征之一，权值对称保证了能量方程在神经元激活时能够单调递减。

离散型 Hopfiled 神经网络的稳定状态包括两个充分条件，一方面属于无自反馈网络；另一方面权值是一个对称矩阵，即无自反馈的权值对称 Hopfiled 神经网络是稳定的网络。稳定状态时有一重要概念：吸引子。吸引子表达的是网络稳定时的状态Y。

离散型 Hopfiled 神经网络的工作方式分为串行工作方式和并行工作方式两种。串行工作方式也被称作异步工作方式，表示在某一时刻t只有某一个神经元的状态发生改变，其他神经元的状态保持不变；并行工作方式也被称作同步方式，表示在某一时刻t所有神经元的状态都发生了改变。

从动力系统的角度来看，系统的平衡稳定状态可以理解为系统的能量函数在系统的不断运动过程中，能量值不断减小，并且最后处于最小值。能量函数对神经网络的意义与损失函数对于神经网络的意义相似，均以优化到最小值为目标。针对离散型 Hopfiled 神经网络引入了李雅普诺夫函数（Lyapunov Function）作为网络的能量函数，Lyapunov 函数在稳定性理论及控制理论中地位非常重要，它是用来证明动力系统或自治微分方程稳定性的系统理论函数，因此离散型 Hopfiled 神经网络的能力函数公式如下：

$$E = (-\frac{1}{2})\sum_i \sum_j w_{ij} y_i y_j - \sum_j x_j y_j + \sum_j \theta_j y_j$$

其中，w_{ij}表示神经元i到神经元j的权值，y_i表示神经元i的输出，x_j表示神经元j的外部输入，θ_j为神经元j的阈值。对上述公式进行转换，即有：

$$E = \sum_{j=1}^{n} \left[\left(-\frac{1}{2}\right) \sum_{i=1}^{n} w_{ij} y_i y_j \right] - \sum_{j=1}^{n} x_j y_j + \sum_{j=1}^{n} \theta_j y_j$$

然后再提取公共项，则：

$$E = \sum_{j=1}^{n} \left\{ \left[\left(-\frac{1}{2}\right) \sum_{i=1}^{n} w_{ij} y_i y_j \right] - x_j y_j + \theta_j y_j \right\}$$

上述公式中的E表示系统整体的能量，根据公式可以看出，对于单个神经元j的能量函数可以表示为如下，系统整体的能量为各个神经元的能量之和。

$$E_j = \left(-\frac{1}{2}\right) \sum_{i=1}^{n} w_{ij} y_i y_j - x_j y_j + \theta_j y_j$$

针对每一个神经元的能量变化ΔE_j可以对E_j求导，按照梯度的类似方式进行计算，并按照能量衰减的方向进行调节，最终使得系统达到稳定的状态。值得说明的是，离散型 Hopfiled 神经网络一般情况下采用的是赫布学习规则，已知在网络中n个神经元相互连接，每个神经元的激活状态为 1，抑制状态为–1。在学习过程中w_{ij}的调节规则是，若神经元i到神经元j同时处于激活状态，则它们之间的连接强度会加强，可以定义为：

$$\Delta w_{ij} = \eta x_i x_j \quad (\eta > 0)$$

连续型 Hopfiled 神经网络在网络结构上与离散型 Hopfiled 神经网络相同，但在工作过程中，所有神经元都随着时间t进行并行更新，网络状态都随着时间连续改变。

在上文介绍中，Hopfield 神经网络神经元的输出是–1 和 1，也可以是 1 或 0，可以根据具体情况进行选择。Hopfield 神经网络实现相对比较困难，应用场景也相对较少，但它当时的提出引起了神经模拟的热潮，Hopfield 网络是神经网络发展历史上的一个重要的里程碑，也是 20 世纪 80 年代神经网络复兴的重要原因之一，并对后续神经网络的发展产生了积极的影响。

6.3 Elman 神经网络

6.3.1 结构组成

Elman 神经网络是由 Jeffrey L. Elman 于 1990 年提出的反馈型神经网络，提出的初衷主要是针对语音处理问题，它是典型的局部回归网络（Global Feed Forward Local Recurrent）。Elman 网络可以被认为是具有局部存储器单元和局部反馈连接的神经网络。

Elman 神经网络的主要结构和前馈型神经网络类似，包括了输入层、隐藏层以及输出层，不同点在于其在传统的隐藏层基础上，新增了一个特殊的隐藏层，一般称作承接层或关联层。承接层从隐藏层中接收反馈信号，每一个隐藏层的神经元都对应一个承接层的神经元节点。关联层的作用是通过联想记忆将上一时刻隐藏层状态以及当前时刻的网络输入一起作为当前隐藏层的输入，类似于状态反馈的机制。对于 Elman 神经网络而言，隐藏层的传递函数一般为非线性函数，例如 Sigmoid 函数；承接层和输出层均为线性函数。Elman 神经网络通常是一个两层的网络结构，其隐藏层神经元与输出层之间存在一个具备反馈连接的承接层。

一个简单的 Elman 神经网络的结构大致如图 6-3 所示，Elman 神经网络的特点是隐藏层的输出通过承接层的存储并自动关联到隐藏层的输入，这种连接方式会使得模型对历史状态的数据比较敏感，而内部承接层的加入也增加了 Elman 神经网络的动态处理能力。

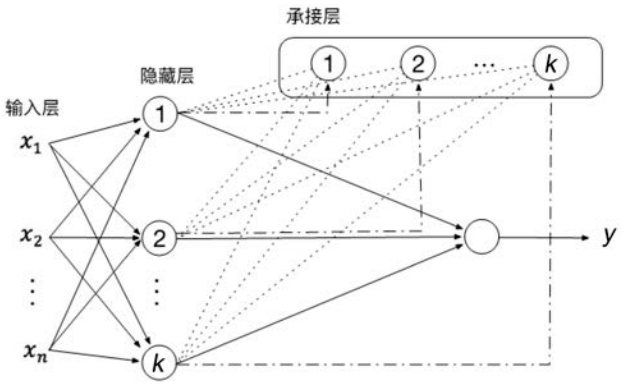


图 6-3 简单的 Elman 神经网络

6.3.2 学习算法

Elman 神经网络的训练过程与前馈型神经网络的学习过程类似，首先初始化各网络层的权值，然后将样本传入输入层并计算输出层输出，再将输入层的输出和承接层的输出传入隐藏层并计算隐藏层的输出，将隐藏层的输出同时传输到输出层和承接层。传递到输出层则计算误差，

然后反向传播调整权值；而传递到承接层则作为下一次隐藏层的输入。

设定 Elman 神经网络的输入层向量为n维的$\boldsymbol{X}$，即$\boldsymbol{X}=[x_1,x_2,x_3,\dots,x_n]^{\mathrm{T}}$；输出层向量为$m$维$\boldsymbol{Y}$，即$\boldsymbol{Y}=[y_1,y_2,y_3,\dots,y_m]^{\mathrm{T}}$；隐藏层的输出向量为$k$维的U，即$U=[u_1,u_2,u_3,\dots,u_k]^{\mathrm{T}}$；承接层的输出向量也是$k$维的$U_c$，即$\boldsymbol{U_c}=[u_{c1},u_{c2},u_{c3},\dots,u_{ck}]^{\mathrm{T}}$；隐藏层到输出层的权值为$w_1$，输入层到隐藏层的权值为$w^2$，承接层到隐藏层的权值为$w^3$；$f$表示承接层的激活函数，一般为 Sigmoid 函数，也可以是高斯函数，根据具体的函数形式，既可以选择反向传播算法调整权值，也可以用 RBF 的方法训练权值，g表示输出层神经元的激活函数，h表示隐藏层神经元的激活函数；t表示神经网络所在的不同时刻；Output表示某一层的输出，用a表示输入层，用b表示承接层。

针对输出层，则存在$y(t+1)=g(\mathrm{Output}_m(t+1))$，而对于$\mathrm{Output}_m(t+1)$，则计算公式如下：

$$\mathrm{Output}_m(t+1)=\sum w^1(t+1)u_k(t+1)$$

针对隐藏层，则存在$u_k(t+1)=f(Output_k(t+1))$，而$\mathrm{Output}_k(t+1)=\sum w^i(t)v_i(t)$，其中$w^i(t)=\begin{cases}w^2,\text{如果 } i\in a\\ w^3,\text{如果 } i\in b\end{cases}$，同理$v_i(t)=\begin{cases}x_n(t),\text{ 如果 } i\in a\\ u_{cn}(t),\text{如果 } i\in b\end{cases}$。

针对承接层，$\mathrm{Output}_n(t)=\sum_{i\in a\cup b}w^i(t-1)v_i(t-1)$，而$u_c(t)=h(\mathrm{Output}_n(t))$，按照上述过程迭代推理即可。值得说明的是，定义的目标函数可以为如下公式：

$$E=\sum_{k=1}^{\mathrm{T}}(y_k-y_k')^2$$

其中，y_k是 Elman 神经网络的实际输出，y_k'表示期望模型的输出。

6.4　递归神经网络

递归神经网络是两种反馈神经网络的总称：一种是时间递归神经网络（Recurrent Neural Network，RNN），另一种是结构递归神经网络（Recursive Neural Network，RNN）。值得说明的是，通常时间递归神经网络被称作循环神经网络，而结构递归神经网络就被简称为递归神经网络。

递归神经网络可以处理变长输入，时间递归神经网络可以处理变长序列信息，结构递归神经网络可以处理变长结构（通常是树形）信息。结构递归神经网络可以看作是时间递归神经网络从时间到空间的泛化。循环神经网络已经发展得比较成熟，可用于包括文本生成、股票预测

等领域，本书第 9 章单独介绍了循环神经网络。结构递归神经网络目前还在发展期，研究成果相对于循环神经网络要少，本节后续的内容将直接用递归神经网络来称谓结构递归神经网络。

递归神经网络早在 20 世纪 90 年代被提出，近年来 Richard Socher 等人基于递归神经网络在自然语言处理领域有深入研究，在多个自然语言处理的任务上有突破性优化：如递归自编码、句法分析、情感分析等。

递归神经网络很灵活，有多个变种，分别在不同的任务中应用并取得了较好效果。

6.4.1 产生背景

传统神经网络的输入层单元个数是固定的，很难处理变长的输入，需要使用各种预处理的技巧将输入向量长度对齐，而且没有考虑输入之间的关系，丢失了部分表达能力，为了解决这些问题，需要构造更复杂的网络。

循环神经网络能处理变长顺序结构，但对于更复杂的树形结构，例如自然语言处理中的语法树结构，循环神经网络略显不足。特别地，在自然语言处理领域中对自然语言的理解上，传统的方法通常有两方面不足。

（1）语言模型简单。最常见的是词袋模型，它将文档分词后统计每个词的出现频率，得到（词，词频）列表。这种处理忽略了词的顺序和语义结构，当用这种模型去理解句子时，会导致明显的问题。比如用词袋模型做情感分析时，词袋中的词汇通过查询情感词典，被标记为正面的或负面的，通过计数来评估整个句子或段落的情感是正面还是负面。针对下面两个句子：

“我不喜欢爬山”
“不，我喜欢爬山”

上述句子移除标点符号后，两个句子的词袋模型是一样的，所以两个句子的情感分析结果也是一样的，均为负面的。但很明显，第二个句子是正面的。另外一个常见的是 N-Gram 模型，它假设当前词只与前面的或者附近的 N 个词有关，这种模型 N 一般不会设得太大，因为计算量会呈指数增长，一般 $n \leqslant 2$，这样的表达能力通常也是有限的。

（2）特征表示困难。一般情况下，自然语言模型或算法的效果非常依赖于语言的特征表示方式。例如，文本分类算法通常要组合多种特征才能达到较好的准确度：词性标注、命名实体、词之间的关系（如 WordNet）。而这些特征需要花费很多时间集成，且很难移植到不同的语言，因为不同的语言则意味着需要重新做一遍类似工作。

递归神经网络有效地解决了上述两个问题，一方面递归神经网络独特的树形结构在语义层面上给语言建模增强了表达能力，另一方面递归神经网络简化了特征选择，减少了前期工作，

具备更强的移植性。循环神经网络在处理较长的序列时容易发生梯度消失，从而无法捕捉长距离的影响。递归神经网络可以在一定程度上解决这个问题，因为对于长度为τ的序列，深度能从$O(\tau)$大幅度降低到$O(\log\tau)$。

尽管递归神经网络具有很强的表达能力，但在目前还处于逐步应用阶段，主要是因为需要预先标注大量的树形结构数据，而且并不是所有领域都有明显的树形结构语义，因为自然语言有方法可以自动标注树形数据，所以目前主要是在自然语言处理领域成功的广泛应用。

6.4.2　基本结构

递归神经网络的典型结构如图 6-4 所示，为简单的单层网络，子节点C_1和C_2为输入，C_1、$C_2 \in \mathbb{R}^d$，将C_1、C_2连接成$2d$维向量，父节点的输出$P_{1,2}$按如下公式计算：

$$P_{1,2} = tanh(\mathrm{W}\times[C_1; C_2] + b)$$

$\mathrm{W} \in \mathbb{R}^{d*2d}, b \in \mathbb{R}^d$，$\mathrm{P}_{1,2} \in \mathbb{R}^d$与子节点$C_1$和$C_2$的维度相同。$tanh$是激活函数，W和$b$在整个递归过程中共享。

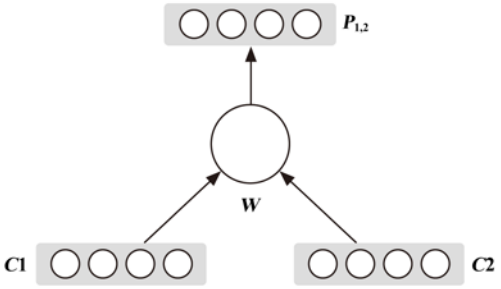


图 6-4　递归神经网络的典型结构

注意，子节点个数可以大于 2 个，假设叶子节点个数为m，非叶子节点的子节点个数为 2，则整个树有$2\times n-1$个节点。

这种网络结构可用于句子的向量空间表示，叶子节点是每个词的向量空间表示，非叶子节点为连续词语组成的短语的向量空间表示，特别地，树的根节点是整个句子的向量空间表示。

在具体的任务中，一般父节点计算得到向量表示后会再有一个目标函数，如图 6-5 中的两个网络结构，与上面的简单单层网络的主要区别在于输出层，计算得到$P_{1,2}$继续作为输入分别得到进一步的输出。

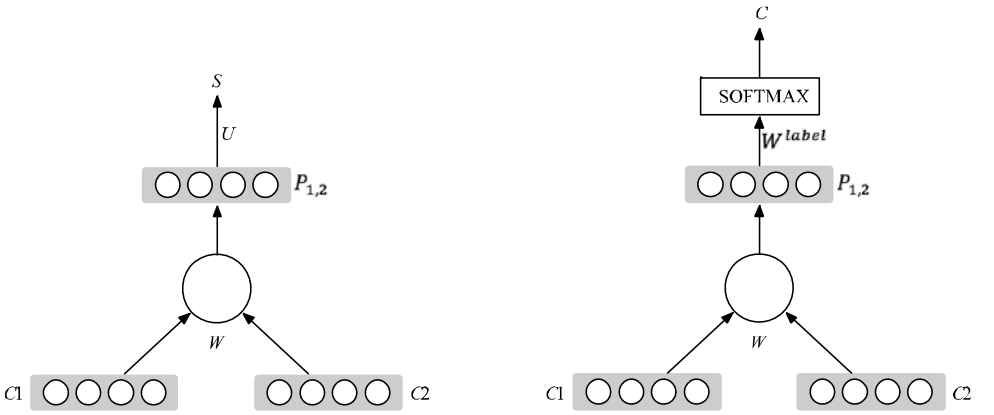


图 6-5　不同目标函数的递归神经网络

图 6-5 的左半部分这个网络用于句法解析任务，输出子节点结合成父节点是否合理的分值S，计算公式为：

$$S = U^{\mathrm{T}} * P_{1,2} + bs$$

其中$U \in \mathbb{R}^{d}$，$bs \in \mathbb{R}^{1}$。

图 6-5 的右半部分这个网络可以用于情感分类任务，经过 softmax 层，输出父节点的情感分类C，计算公式为：

$$C = \text{softmax}（W^{\text{label}} * P_{1,2}）$$

6.4.3　前向计算过程

递归神经网络的前向计算过程则是从下而上深度优先遍历树结构，计算每个节点的输出，再将它作为其父节点的输入，计算其父节点的输出，直至树的根节点。假设是二叉树结构，则整个递归过程可以描述为：

1. 计算左子节点输出，如果是叶子节点，则输入即输出。
2. 计算右子节点输出，如果是叶子节点，则输入即输出。
3. 计算父节点输出。

以图 6-6 所示的网络结构为例，介绍递归神经网络的整个前向计算的过程，已知每个叶子节点的向量空间表示与语法树，通过该网络结构对应的前向计算公式得到 Node1（我爱）、Node2（北京天安门）的向量空间表示和结合分值$S1$、$S2$。再将 Node1、Node2 作为子节点，计算 Node3（我爱北京天安门）的向量空间表示和结合分值$S3$。

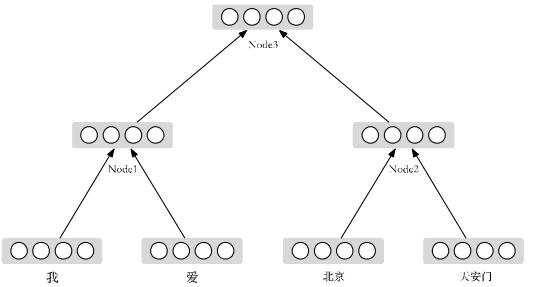


图 6-6　递归神经网络的前向计算

6.4.4　反向传播：BPTS 算法

递归神经网络的反向传播算法（Back Propagation Through Structure，BPTS）与循环神经网络的反向传播算法 BPTT 类似，循环神经网络需要将残差从当前时刻t_k传递到初始时刻t_1，递归神经网络需要将残差从根节点反向传播到各个子节点。

定义误差项δ^p为误差函数E相对于父节点p的加权输入的倒数，即：

$$\delta_p = \frac{\partial E}{\partial \mathbf{net}_p}$$

$$\boldsymbol{\delta}_c = \left(\boldsymbol{W}^{\mathrm{T}}\delta_p\right) \circ f'(\mathbf{net}_c)$$

其中$\boldsymbol{\delta}_c = \begin{bmatrix}\delta_{c1}\\ \delta_{c2}\end{bmatrix}$，$\boldsymbol{net}_c = \boldsymbol{W}\begin{bmatrix}C_1\\ C_2\end{bmatrix} + b$。从根节点开始计算误差项，深度遍历能计算出所有节点的误差项。l层的梯度计算公式为：

$$\frac{\partial E}{\partial \boldsymbol{W}^{(l)}} = \delta_p^{(l+1)}\left(\boldsymbol{a}^{(l)}\right)^{\mathrm{T}}$$

其中$a^{(l)} = \begin{bmatrix}C_1\\ C_2\end{bmatrix}^{(l)}$，$\delta_p^{(l+1)}$为父节点的误差项。类似于误差项的计算过程，深度遍历计算所有节点的梯度，并通过梯度下降算法更新权值。

递归神经网络的反向传播算法有以下三个特点。

（1）由于权值矩阵$\boldsymbol{W}$是所有层共享的，因此递归神经网络的误差是所有非叶子节点的误差之和。

（2）在前向计算过程中，C_1、C_2组合之后再计算父节点的输出，所以反向传播时也应该分别计算：

$$\boldsymbol{\delta}_{p\to C1C2} = [\boldsymbol{\delta}_{p\to C1}\boldsymbol{\delta}_{p\to C2}]$$

（3）每个节点的误差来自于两部分，一部分是父节点反向传递过来的误差；另一部分是目标函数的误差。

6.4.5 应用场景

目前递归神经网络在自然语言处理中有较深入的应用，因为自然语言处理天然具备语义上的树形结构，目前递归神经网络已广泛应用在这一领域的多个任务中，主要的任务和对应的网络结构如表 6-1 所示。

表 6-1 自然语言处理任务与递归神经网络对应表

任　　务	网络结构
句法解析	朴素递归神经网络（Standard RNNs）
关联分类	矩阵—向量递归神经网络（Matrix-Vector RNNs）
情感分析	递归神经张量网络（Recursive Neural Tensor Networks）
短语相似性	树形长短时记忆网络（Tree LSTMs）

表中的几个非朴素递归神经网络是朴素递归神经网络的变种，其实表 6-1 中所有网络模型都可以应用到表中的任务，但是对于特定的任务，相对应的网络会表现得更好。所以如果想要获得好的效果，针对新的任务，可以对网络进行针对性的改进。

6.4.6 递归神经网络的结构改进

经前文的介绍，朴素递归神经网络存在两方面的问题。一方面权值矩阵$\boldsymbol{W}$是全局共享的，会导致网络的表达能力不足；另一方面权值矩阵$\boldsymbol{W}$可以看作是两个矩阵的拼接，子节点C_1、C_2其实相互没有发生关系。

针对朴素递归神经网络的两方面问题对递归神经网络进行结构上的改进，分别是语义解绑递归神经网络、矩阵—向量递归神经网络、递归神经张量神经网络。

1. 语义解绑递归神经网络

所有的子节点合并都用同一个权值矩阵$\boldsymbol{W}$，直观上可能会影响模型的表现，但如果所有的子节点合并的权值矩阵$\boldsymbol{W}$都不同，则会使得要学习的矩阵暴增。可以考虑按照某种分类来共享权值矩阵$\boldsymbol{W}$，一方面增强了表达能力，另一方面又控制了要学习的矩阵数量。

以在向量空间模型上表示句子的语义这一任务为例，考虑根据子节点的句法类型来分类共享权值矩阵W，句法类型是指词性组合，例如名词后跟着动词。可以通过概率上下文无关文法（Probabilistic Context Free Grammar，PCFG）来确定每个子节点的句法类型。

一个典型的语义解绑递归神经网络结构如图 6-7 所示，每个节点由（词性和词向量）组成，

每个父节点的权值矩阵$\boldsymbol{W}^{(\text{Left,Right})}$由左右子树的句法类型来决定。

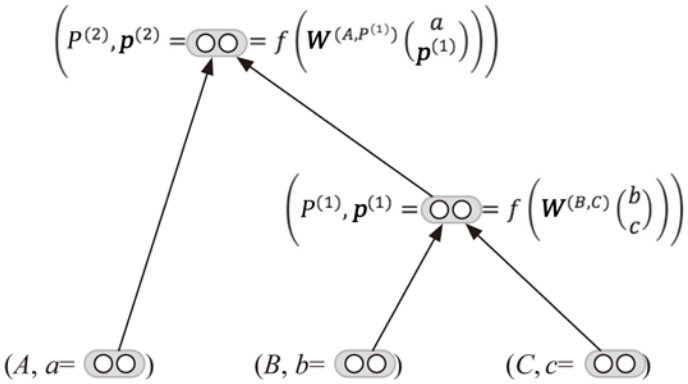


图 6-7　语义解绑递归神经网络

当前模型的另一个改进点是将权值矩阵$\boldsymbol{W}$初始化为单位矩阵，而非随机值。训练之初，$\boldsymbol{W}$会对输入的两个词向量进行均值计算，但经过一段时间训练后会发现，模型学习到了词向量对于句子的重要性，句子中更重要的词语会被呈现出来。例如，对于限定词—名词、冠词—名词这些句法类型，名词的权值会明显高于限定词和冠词。这一语言现象被称作“软头词”（Soft Head Word），语言学专家通过长时间的研究才发现这一概念，模型也可以通过训练学习到该特征。

语义解绑递归神经网络相对于朴素神经网络，解决了问题权值矩阵$\boldsymbol{W}$是全局共享的问题，但还是没有解决子节点之间没有互相发生关系的问题，但表达能力已经有所提升。

2. 矩阵—向量递归神经网络

前面介绍的语义解绑递归神经网络没有解决子节点之间没有互相发生关系的问题。例如，针对句子：“这部电影很好看”，其中“很”为副词，“好看”为形容词，“很”的字面意思是“用于强调”，之前介绍的所有模型都不能将这种强调关系影响到“好看”，所以需要考虑有没有办法使得一个词向量可以强调另一个词向量，即对另一个词向量有“缩放”的效果。通过增加一个词矩阵给子节点之间建立这种“缩放”关系。矩阵—向量递归神经网络的结构如图 6-8 所示。

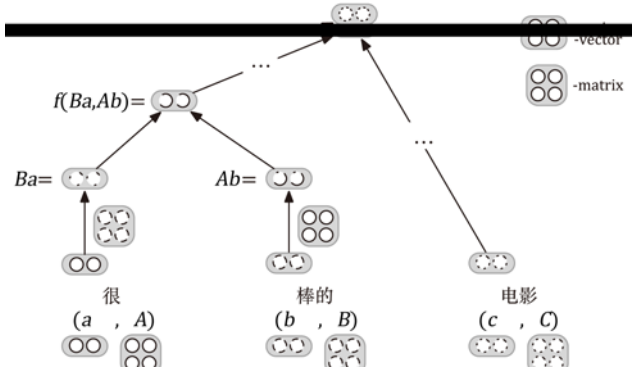


图 6-8 矩阵—向量递归神经网络结构示例

因此每个词由词向量和词矩阵组成，词向量$\boldsymbol{V} \in \mathbb{R}^{d}$，词矩阵$\boldsymbol{M} \in \mathbb{R}^{d*d}$，左子节点输入为右子节点词矩阵乘以左子节点词向量，右子节点输入为左子节点词矩阵乘以右子节点词向量，再根据这两个输入计算得到父节点的词向量，计算公式为：

$$\boldsymbol{V}_p = \boldsymbol{W} * (\boldsymbol{M}_r * \boldsymbol{V}_l; \boldsymbol{M}_l * \boldsymbol{V}_r) + b$$

“很”节点的词矩阵可以是单位矩阵乘以大于 1 的常量k，因此可以将相邻节点的词向量放大k倍，这就解决了子节点之间没有互相发生关系的问题，将子节点之间建立了相互关系。虽然增加了词矩阵导致模型需要学习的参数增多，但是这种新的模型表达能力更强。

3. 递归神经张量网络

矩阵—向量递归神经网络具有大量的参数，而且参数数量随词汇表的大小增长而增长。自然会想到是否有一个固定参数数量的强大的单一组合函数能具备同样的表达能力。递归神经张量网络（Recursive Neural Tensor Network，RNTN）则是这样一个具备强大组合函数的网络。RNTN 在解决权值W全局共享导致表达能力不足以及子节点没有产生相互关系问题的同时，降低了整体的计算量。

如图 6-9 所示，为单一的张量层。

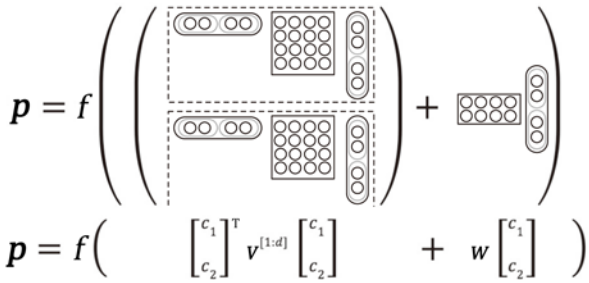


图 6-9 递归神经张量网络的示意图

我们定义输出张量$\boldsymbol{h}$计算公式为：

$$\boldsymbol{h}=\begin{bmatrix}c_1\\c_2\end{bmatrix}^{\mathrm{T}}\boldsymbol{V}^{[1:d]}\begin{bmatrix}c_1\\c_2\end{bmatrix};\boldsymbol{h_i}=\begin{bmatrix}c_1\\c_2\end{bmatrix}^{\mathrm{T}}\boldsymbol{V}^{i}\begin{bmatrix}c_1\\c_2\end{bmatrix}$$

其中$\boldsymbol{h}\in\mathbb{R}^d$，$\boldsymbol{V}^{[1:d]}\in\mathbb{R}^{2d\times 2d\times d}$。输出的组合函数为：

$$p=f\left(h+\boldsymbol{W}\begin{bmatrix}c_1\\c_2\end{bmatrix}\right)$$

递归神经张量网络可以较好地判别自然语言中的否定情感词，是目前递归神经网络在情感分析任务中表现最好的网络结构。

6.4.7　应用实例

可以从自然语言处理中的句法解析、无监督递归自编码以及情感分类的应用中深入了解递归神经网络。

1. 句法解析

自然语言可以通过句法树来表示内部的结构信息，因此可以把从句子到句法树的过程称为句法解析，句法解析是自然语言处理的基础任务之一，可以帮助解决其他任务，包括句子分类（Sentence Classification）、机器翻译（Machine Translation）、语义角色标注（Semantic Role Labeling）等。句法解析有很多种方法，最为常用的是依存句法解析（Dependency Parsing）和基于短语的句法解析（Constituency Parsing）。

通过递归神经网络做句法解析的好处在于不用提取复杂特征和设计特定规则。网络结构如图 6-5 中的左图部分所示，训练集为标注好的句子以及对应的句法树，训练过程为通过多次迭代的前向计算、反向传播，得到权值矩阵。假设现在已经训练好了递归神经网络的参数，只考虑如何构造句子的句法解析树。我们用柱搜索这一算法来构造句法解析树，整个过程可以描述为如下：首先对于当前的所有子节点，相邻节点两两计算结合分值；然后选取结合分值中最高的相邻节点结合成一个节点；不断重复前面步骤，直至只剩下一个节点。

以句子“我爱北京天安门”为例，计算过程如图 6-10 所示。首先两两计算结合分值，（“我”，“爱”）的结合分值最高，将它们组合成“我爱”节点，再计算“我爱”、“北京”、“天安门”两两的结合分值，（“北京”，“天安门”）的结合分值最高，将它们组合成“北京天安门”，现在只剩下两个节点了，组合起来并计算根节点的向量表示，即这个句子的向量表示。

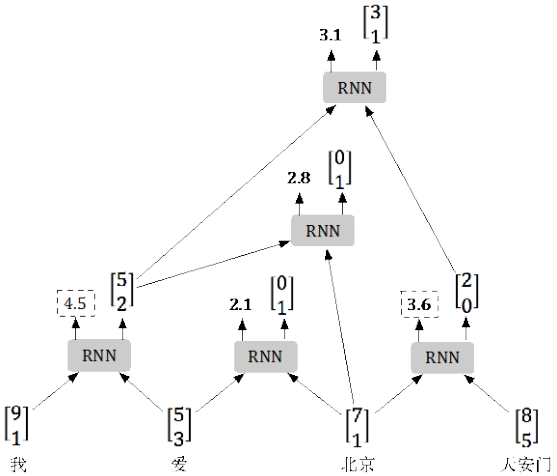


图 6-10　柱搜索句法解析过程

2. 无监督递归自编码

首先讨论递归神经网络在自然语言处理中最基础的应用：在向量空间模型上表示句子的语义，后续的任务很多是在这个基础上进行的，这一任务通常称为递归自编码。自然语言处理中通常用一个向量来表示一个词，好处在于能很好地度量词与词之间的相似性，在向量空间中越近则表示越相似，进一步也可以考虑用一个向量来表示一个句子或段落，用于度量句子或段落之间的距离。例如下面两个句子：

我出生的国家
我出生的地方

两个句子表达的语义应该是相近的，它们在向量空间中的距离应该也是相近的，可以通过复合型原理来计算句子的向量：一个复杂表达式的意义是由其各组成部分的意义以及用以结合它们的规则来决定的。一个句子的语义来自于所有词的语义以及所有词结合成句子的规则，其中所有词的语义为词向量，所有词结合成句子的规则为句法解析树。

词嵌入是自然语言处理中语言模型与表征学习技术的统称，有时也被称为词向量。概念上而言，它是指把一个维数为所有词的数量的高维空间嵌入到一个维数低得多的连续向量空间中，每个单词或词组被映射为实数域上的向量。目前已经存在一些比较好的词嵌入工具，例如Word2Vec。

为了得到句子的向量，可以通过自编码器获得。自编码器是一个浅层的神经网络，是由Hinton 等人在 1989 年提出的一个概念，可以理解为一个尽可能去复现原始输入的系统。原始输入通过编码、解码重构后输出，目标是让这个输出尽可能地与输入相同。训练的过程就是调

整参数，得到每一层的权值，每一层就可以认为是输入的一种表示。通常中间层的节点个数会比输入要少，这样就能像主成分分析（Principal Component Analysis，PCA）一样提取出主要特征。自编码器是无监督的，可以使用自编码器来计算句子的向量表示。

句子的向量表示这一任务中的自编码器结构如图 6-11 中实线框内部分所示，输入层有 2 个节点，中间层只有 1 个节点，即左右子树上的短语结合后的表示，输出层仍为 2 个节点，预期输出与输入相同。

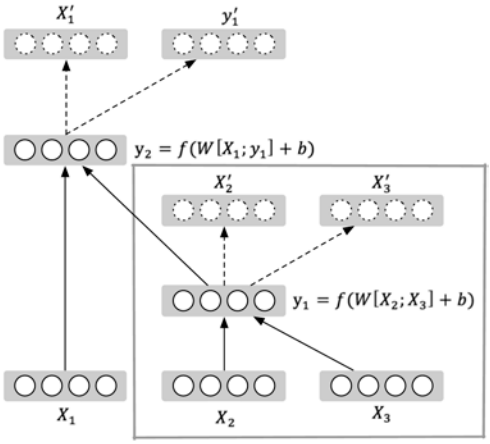


图 6-11　递归自编码网络结构

递归自编码与普通的自编码器的区别在于重构误差函数，普通的自编码器的重构误差即输出与输入之间的误差，而递归自编码需要考虑整个树形结构产生的误差。假设一个句子x用$(x_1, x_2, \ldots x_m)$来表示，句子长度为m，第k个词为x_k。用$A(x)$表示句子x所有可能的树形结构集合，用$T(y)$表示$A(x)$中某个树形结构y中所有的非叶子节点的集合，那么目标重构误差函数为：

$$RAE_\theta(x) = \arg\min_{y \in A(x)} \sum_{s \in T(y)} E_{\text{rec}}([c1; c2]_s)$$

通过不断迭代，使得所有句子的$RAE_\theta(x)$之和小于某个阈值则停止迭代，对于句子x，选择$A(x)$中重构误差最小的那个树形结构y，y的根节点输出即为句子的向量表示。

3. 情感分类

情感分类是自然语言处理中的一个被广泛研究的任务。无监督递归自编码实例中，如果给每个句子标注情感类别，那么则可以做半监督的情感分析。但是这种用贪心算法来构建句法树需要巨大的运算量，并且无监督的递归自编码 RAE 忽略了词语本身的语序对情感的影响。另外，Socher 等人发现 RAE 在情感分析 task 中考虑重构误差意义不是特别大，他们提出了一种基于 TreeBank 的递归神经张量网络。TreeBank 是斯坦福大学标注的一个情感分析语料集，格

式如下：

(1(2 Today)(((2 is)(1 not))((2 a)(3 good)(2 day)))(2 .))

标注信息包含句法树结构，以及句法树每个节点所代表的词语或短语的情感分类，0 表示强烈的负面，1 表示负面，2 表示中立，3 表示正面，4 表示强烈的正面。

组合函数使用 RNTN，目标函数使用如图 6-5 右边部分的 Softmax，经过 RNTN 的训练后，可以预测句子的情感，此方法在 TreeBank 语料集上，统计所有节点的准确率可达到 87.6%，统计根节点的准确率可达到 85.4%。

6.5 本章小结

本章从各个方面详细介绍了反馈型神经网络，它是一种从输入到输出具有反馈连接的神经网络结构，其结构比前馈型神经网络复杂。本章详细介绍了几个典型的反馈型神经网络：Hopfiled 神经网络、Elman 神经网络，以及递归神经网络。

Hopfiled 神经网络又分为离散型 Hopfiled 神经网络和连续型 Hopfiled 神经网络，主要区别在于两者的输入输出是离散型或连续型，Hopfiled 最早提出的 Hopfiled 神经网络属于离散型，在离散型的基础上继而提出了连续型 Hopfiled 神经网络。Elman 神经网络是两层反向传播网络，最大的结构差异在于隐藏层和输入层之间还包含了承接层，承接层的存在是一种反馈机制，也反映出 Elman 神经网络的时序特征。递归神经网络可以处理变长的输入，包括时间上的顺序序列和空间上的树形结构，是一种比较复杂的反馈型神经网络，但也成功地在多个领域里广泛应用。

除 Hopfiled 神经网络、Elman 神经网络、递归神经网络外，反馈型神经网络还包括海明神经网络、双向联想存储器网络等。

第 7 章

自组织竞争型神经网络

自组织神经网络，也称为自组织竞争型神经网络，特别适用于解决模式分类和识别的问题。网络模型属于前向神经网络模型，采用无监督学习算法。其工作的基本思想是让竞争对手的神经元通过竞争和输入模式相匹配。最后只有一个神经元成为比赛的冠军，获胜神经元的输出表示输入模式的分类。

7.1 概述

一般在传统的机器学习领域中常常涉及无监督的算法，大量数据的计算通过“无师自通”的方式，“无师自通”的核心在于能够对数据之间的内在关系进行反复观察、分析、比较，发现数据集中的内在规律，并通过数据之间的特征将数据正确归类，以达到无监督的目的。在神经网络中，自组织竞争型神经网络也是通过发现数据的内在规律和本质属性，并有效的自组织、自适应地改变网络参数和结构。

常用的自组织竞争型神经网络包括自适应共振理论网络（ART）、自组织特征映射（SOM）网络、传输（CP）网络和协同神经网络（SNN）等，它们均属于无监督式学习网络（Unsupervised Learning Network），其主要是对原始数据进行自动化聚类，以便分析数据之间的特征。不同于监督式学习网络，无监督式学习网络在学习时并不知道其分类结果是否正确，不存在人工标记的数据确定其分类正确与否。其特点是仅对此种网络提供输入范例，而它会自动从这些范例中找出其潜在类别规则。当学习完毕并经测试后，也可以将其应用到新的案例上。

7.1.1 一般网络模型

一般而言，自组织竞争型神经网络的网络结构主要分为输入层、竞争层、输出层，大致结构如图 7-1 所示。

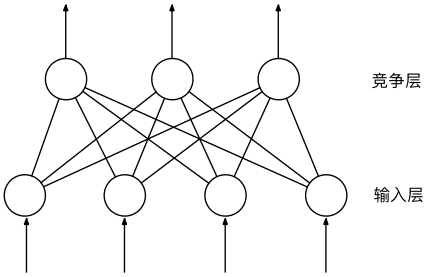


图 7-1 自组织竞争型神经网络的一般模型示例

输入层负责接收外界数据信息，并将输入模式向竞争层传递，属于数据的“观察者”角色；竞争层则按照竞争规则更新权值，每次仅更新在竞争中获胜的神经元权值。竞争层与输入层属于全连接的关系，并且竞争层内神经元之间不存在任何连接，竞争层负责对输入层传入的模式进行分析比较，找出数据之间的规律，以将数据正确归类。输出层的输出则为线性函数，将训练好的权值线性输出。

7.1.2 工作原理

在本书的神经网络基础章节已经介绍了竞争学习规则，竞争学习规则是自组织竞争型神经网络的基础。采用竞争学习规则可以构成最简单的自组织竞争型神经网络。在竞争学习规则的基础上，还衍生出其他各式各样的自组织竞争型神经网络。由于网络中的输出神经元之间相互竞争以期望在网络中被激活，但最终结果是每一时刻仅有一个输出神经元被激活，因此这样的竞争学习规则也被称作“Winner Take All”。

“胜者为王，败者为寇”是自组织竞争型神经网络的核心原理。激活状态最强的神经元通过竞争的方式战胜其他神经元，从而在权值调整中其激活状态会被进一步加强，而竞争中失败的神经元状态则保持不变。自组织竞争型神经网络通过这样的竞争学习规则，获得训练样本的分布情况，最终每个训练样本都存在一个处于激活状态的神经元，而这个神经元实则对应一个样本类别，当有新的样本作为输入时，则可以根据神经元的激活状态或抑制状态进行模式分类。

在自组织竞争型神经网络中，与传统的神经网络有一个输入的差异，即模式。模式在广泛的含义中是指存在于时间和空间中可以被观察的物体，如果区分两个物体是否相同或相似，则可以称之为模式。模式并不是事务本身，而是指从事务中获取的信息，而信息一般包括时间信息和空间的分布信息。模式并不是自组织竞争型神经网络中的概念，在模式识别、模式分类中也存在。

自组织竞争型神经网络的一般工作流程同竞争学习规则的流程基本一致，即：

1. 向量归一化。

② 寻找获胜的神经元。

③ 权值调整。

7.1.3　实例：用竞争学习规则进行模式分类

现有一堆二维坐标数据如下所示，可以根据竞争学习的方法将模式分为两个类别：

$x_1 = (0.87,0.50)$

$x_2 = (0.40,0.92)$

$x_3 = (0.20,-0.98)$

$x_4 = (0.1,1.0)$

$x_5 = (0.3,-0.96)$

同二维坐标系一样，极坐标系也存在两个坐标轴r与θ，r表示半径坐标，是与极点之间的距离；θ表示角坐标，表示按照逆时针方向与极轴（0°射线）的角度。极轴是平面直角坐标系中 x 轴的正方向。直角坐标转极坐标的转换公式如下：

$$r = \sqrt{x^2 + y^2}$$

$$\tan\theta = \frac{y}{x}$$

相反，极坐标转换为直角坐标公式：

$$x = r\cos\theta$$

$$y = r\sin\theta$$

极坐标中的点可以用(r,θ)表示，例如（2,45°）表示距离极点 2 个单位长度，和极轴的夹角为45°。

针对上述给出的输入模式，可以将其转换为极坐标形式，转换后的极坐标如下：

$x_1 = (1.0, 29.89^\circ)$

$x_2 = (1.0, 66.50^\circ)$

$x_3 = (1.0, -78.47^\circ)$

$x_4 = (1.0, 84.29^\circ)$

$x_5 = (1.0, -72.65^\circ)$

根据上述的极坐标，可以绘制模式向量图，如图 7-2 所示。

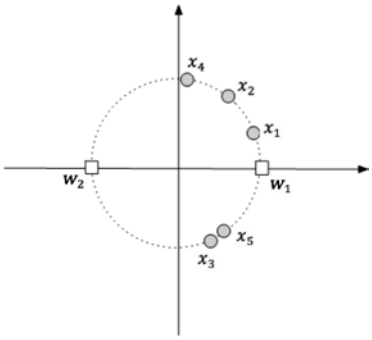


图 7-2　模式向量图

将模式分为两类，则说明竞争层存在两个神经元，因此设置两个权向量$\boldsymbol{w}_1$和$\boldsymbol{w}_2$，并初始化权向量$\boldsymbol{w}_1(0)=(1.0,0^{\circ})$、$\boldsymbol{w}_2(0)=(1.0,-180^{\circ})$，$\boldsymbol{w}_1(i)$表示在第$i$次迭代时权向量的值，权向量$\boldsymbol{w}_1$和$\boldsymbol{w}_2$的位置如图 7-2 所示，设定$\eta$表示学习速率，默认为 0.8。因此，采用竞争学习将模式分为两类的步骤如下。

① 首先以模式x_1作为输入，分别计算x_1与$\boldsymbol{w}_1(0)$、$\boldsymbol{w}_2(0)$的距离。

$$d_{x_1w_1}=\left||x_1-\boldsymbol{w}_1(0)|\right|=29.89^{\circ}$$

$$d_{x_1w_2}=\left||x_1-\boldsymbol{w}_2(0)|\right|=209.89^{\circ}$$

显然，$d_{x_1w_1}<d_{x_1w_2}$，则神经元 1 获得胜利，因此需要对$\boldsymbol{w}_1$的权值进行调整，公式如下：

$$\boldsymbol{w}_1(1)=\boldsymbol{w}_1(0)+\eta\big(\boldsymbol{x}_1-\boldsymbol{w}_1(0)\big)=(1.0,23.912^{\circ})$$

而神经元 2 属于竞争失败，因此$\boldsymbol{w}_2(1)=\boldsymbol{w}_2(0)=(1.0,-180^{\circ})$。

② 按照第一步的方式，依次把模式x_2、x_3、x_4、x_5作为输入，不断选择获胜的神经元，并调整相应神经元的权值。

③ 当不断迭代上述步骤之后，$\boldsymbol{w}_1$和$\boldsymbol{w}_2$的值变化会不断趋于稳定，$\boldsymbol{w}_1$的值趋近于$(1.0,60^{\circ})$，$\boldsymbol{w}_2$的值趋近于$(1.0,-75^{\circ})$，x_1、x_2、x_4属于同一类模式，而x_3、x_5属于同一类模式。

值得说明的是，学习速率η如果一直为常数，则$\boldsymbol{w}_1$和$\boldsymbol{w}_2$的值会一直处于某个区间不断摆动的位置，为了解决这类区间摆动的问题，一般采用η作为变量的形式，并随着迭代次数的增加，η的值逐渐变小，使得模型尽可能收敛到最佳位置。

7.2　常见的聚类方法

自组织竞争型神经网络常常用于对数据的自适应聚类，常见的聚类方法包括系统聚类法、基于划分的聚类算法、基于密度的聚类算法和基于层次的聚类算法等。

7.2.1　系统聚类法

系统聚类法（Systematic Luster Method）又被称作层次聚类，系统聚类法是通过计算将距离较近的样本先聚成一类，距离较远的样本后聚成一类，通过不断计算样本之间的距离，最终每一个样本都能找到合适的聚簇。

从聚类的过程分析上，可以将聚类划分为系统聚类法、逐步聚类法、有序样品聚类法、模糊聚类法以及分隔聚类法等。

（1）系统聚类法。主要用于对小数据量的样本之间聚类以及对指标聚类。

（2）逐步聚类法。逐步聚类法也被称作快速聚类法，主要用于对大数据样本之间的聚类。它首先定义了样本聚簇的中心点，这些初始中心点既可以是随机产生的，也可以是通过一定规则产生的。然后对其他样本数据与初始化的中心点进行距离计算，样本数据离中心点越近，则该样本数据越属于该中心点所在的聚簇，然后通过聚簇内的距离计算更新当前聚簇的中心点，不断迭代上述过程直到聚簇中心不再改变。

（3）有序样品聚类法。用于对有序的数据样本进行聚类，即把次序相邻的样本聚为一类的方法。

（4）模糊聚类法。基于模糊数学的样本聚类分析方法，主要适用于小数据样本。

距离是进行系统聚类分析中需要考虑的核心点，由于距离的定义不一样，所以系统聚类法对应的方法也有多种。在聚类中，主要的距离计算方法包括：最短距离法（Single Linkage Method）、最长距离法（Complete Linkage Method）、中间距离法（Median Method）、重心法（Centroid Hierarchical Method）、离差平方和法（Ward Method）以及类平均距离法（Average Linkage Method），这些距离的定义方法包括了前面介绍过的欧氏距离、马氏距离、余弦相似性等，应根据应用场景的不同，选择合适的距离计算方法。

（1）最短距离法。倘若有两个聚类分别为 A、B，则两个聚类间的最短距离可以定义为：$D_{A,B}=\min(d_{i,j})$，其中 $i\in A$，$j\in B$，当计算好最短距离之后，将最短距离的样本点进行合并，不断迭代构造可能的聚类。将两个聚簇之间的距离定义为两个聚簇样本距离的最小值，这种方法不限制聚簇的形状，对于条形分布的数据聚类效果较好。

（2）最长距离法。将两个聚簇之间的距离定义为两个聚簇样本距离的最大值。

（3）中间距离法。将两个聚簇之间的距离定义为两个聚簇样本距离的中间值，是对聚簇之间间距的最长距离、最短距离、聚簇内距离的加权体现。

（4）重心法。将两个聚簇之间的距离定义为两个聚簇样本距离的两个聚类的中心距离，可以较好地处理一些异常值。

（5）类平均距离法。将两个聚簇之间的距离定义为两个聚簇样本距离的平均距离，从结果上体现而言，它倾向于优先合并方差较小的聚簇。

（6）离差平方和法。它的聚类过程中，将合并后聚簇之间距离方差最小的聚簇合并，倾向于产生数据相近的两个聚类，对异常值较为敏感。

7.2.2 基于划分的聚类算法

常用的划分聚类算法有 K-Means，K-Medoids 以及它们的变种。

K-平均聚类算法（K-Means）是一种无监督的机器学习算法，与 K-近邻之间（KNN）没有任何关系。K-Means 的核心思想是：人以类聚，物以群分；而 KNN 的核心思想是：少数服从多数。

K-Means 作为文本聚类算法最直接的算法，也是最为经典的数据挖掘算法，是一种基于自下而上的聚类分析方法。其基本概念是：空间中有 N 个点，初始选择 K 个点，作为中心聚类点，将 N 个点分别与 K 个点计算距离，选择离自己最近的作为自己的中心点，不断地更新中心聚集点。K-Means 解决的问题如图 7-3 所示。

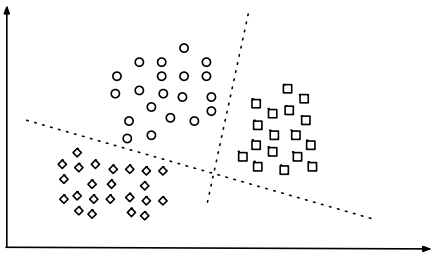


图 7-3 K-Means 的聚类示例

K-Means 算法能够对图 7-3 中零散的数据分布进行划分，并切分为严格的几个聚类，聚类结果如图 7-3 所示，以达到“物以类聚，人以群分”的效果。K-Means 的 K 表示聚类的个数，在一开始即为一个固定值。

K-Means 找到的中心一定是自然聚簇的中心，这点与 EM 算法雷同。其具体算法流程如下。

① 从数据点集合中，随机选择 K 个点，作为初始的聚集中心。每一个中心点，代表着每一个聚集中心的平均值。

② 对其余的数据点，依次判断它与 K 个中心点的距离，距离最近的，则表明它属于这项聚类。

③ 再重新计算新的聚簇集合的平均值即中心点。整个过程不断迭代计算，直到达到预先设定的迭代次数或中心点不再频繁波动。

输入数据：需要聚类的个数 K，包含 N 个数据对象的集合 G。

输出数据：包含 N 个数据对象的 K 个聚类集合。

计算方法：从数据集合 G 中选择任意的 K 个数据作为最初始的聚集中心点；计算出所有数据中与中心点的距离，选择最近的点，作为自己的新聚类。形成新的聚簇集合之后，再计算出 K 个聚簇的中心平均值，确定中心点位置。然后不断迭代上述过程，直到达到预先设定的迭代次数或者聚簇中心不再改变。

7.2.3　基于密度的聚类算法

基于密度的方法与其他方法的一个根本区别是：它不是基于各种各样的距离的，而是基于密度的，这样就能克服基于距离的算法只能发现超球形簇的缺点。这个方法的指导思想是，只要一个区域中的点的密度大过某个阈值，就把它加到与之相近的聚类中去。代表算法有：DBSCAN 算法、OPITICS 算法等。

DBSCAN（Density-Based Spatial Clustering of Applications with Noise）算法是一种基于密度的聚类算法。在 DBSCAN 算法中，预先定义两个变量：一个是 eps 表示半径范围，另一个是 MinPts 表示半径内指定的点的数量，还包括直接密度可达、密度可达、密度相连三个概念，分别如下。

（1）直接密度可达。对于样本集合 N，给定样本点 q，如果 q 在核心点 p 的半径范围内，则称为数据点 q 与核心点 p 直接密度可达。

（2）密度可达。给定样本集合 N 中对象链 $P1$、$P2$、$P3\cdots Pn$，如果 $P1=q$，$Pn=p$，且 $Pi+1$ 在核心点 Pi 的半径范围内，则 p 与 q 密度可达。

（3）密度相连。数据集合中，如果数据点 A 与其中的数据点 P 和数据点 Q 都是密度可达，则认为 P 和 Q 密度相连。

DBSCAN 算法的思想简单而言则是在以某点为核心点的基础上，若在指定半径范围内拥

有超过指定的点数量，则形成一个聚簇。DBSCAN 的点被分为核心点、边界点和噪声点。

（1）核心点。在该点的指定半径范围（eps）内有超过指定数量（MinPts）的点，属于密度稠密区内部的点。

（2）边界点。在指定半径范围内且附近数量小于 MinPts，但是却落在某核心点的领域内，属于密度稠密区边缘上的点。

（3）噪声点。不在核心点的领域内，且在指定半径范围内也不存在达到 MinPts 数量的点，即核心点和边界点之外的点。

DBSCAN 算法的计算流程：首先将所有需要聚类的数据标记为未访问；其次是随机选择一个数据对象 p，将 p 标记为已经访问，如果 p 的半径范围之内存在 MinPts 个对象，则认为 p 是核心点，相应创建一个聚类簇 C，将 p 添加入 C；然后将 p 半径范围内的所有对象设置为 N，对其中的密度相连的点进行遍历。如果是未访问的数据点，则将该数据点标记为已访问；如果该数据点在半径范围内存在的数据对象不少于 MinPts 个，且该数据点还不属于任何聚类簇，则可将此数据点视为是以 p 为核心点的聚类簇 C 的成员。依次遍历，直到最后所有的点都被标记为已访问。在访问过程中，如果 p 不满足在半径范围内数据对象不少于 MinPts 个，则认为 p 为噪声点。简单而言，即判断输入点是否为核心对象，找出核心对象 E 领域中的所有直接密度可达点，重复上述过程直到所有的输入点都判定完毕。

K-Means 和 DBSCAN 都是常见的聚类算法，二者的区别在于 DBSCAN 不需要提前确定簇类的数量，而且 DBSCAN 可以发现任意形状的聚类。但是 DBSCAN 也存在不足，总体缺陷主要集中在两方面：一方面是对于高维数据，不能很好地反映；另一方面是在聚类密度不断变化的数据集中，不能很好地反映整体聚类情况。

为了解决 DBSCAN 算法的全局密度参数不能刻画其内在结构这个问题，提出了 OPTICS（Ordering Points to Identify the Clustering Structure）聚类分析方法。它并没有显式地产生一个数据集合簇，而是为自动和交互的聚类分析计算一个簇次序。这个次序代表了数据的基于密度的聚类结构。它包含的信息，等同于从一个宽广的参数设置范围所获得的基于密度的聚类。

7.2.4 基于层次的聚类算法

基于层次的聚类算法是非参数的方法，主要是通过逐步合并数据点或切分超聚类，从而把数据分组为一棵层次树的方式获得聚类。

基于层次的聚类算法具体可以分为凝聚（agglomerative）和分裂（divisive）两种形式，前一种是自下而上的，后一种则是自上而下的。前者，即凝聚的方法是建立在聚类间距离尺度基

础上的。最初可以把每个数据点看作一个聚类，然后通过把最邻近的聚类融合起来以降低聚类的数量。重复这个过程直到最终只有一个包含所有数据的聚类。而分裂的方法和凝聚的方法正好相反，它是从一个由所有数据组成的聚类开始，然后把这个聚类尽量分裂，重复这个过程，直到满足条件为止（可以一直分裂到一个聚类就只有一个数据）。分裂的方法也有两种类型，即单分裂（Monothetic）和多分裂（Polythetic）。单分裂是指每次用数据的一个属性来分裂聚类，而多分裂的方法则是对全部属性进行综合分析来分裂数据。

一个纯粹的层次聚类方法虽然简单，但却常有以下问题。

（1）合并点或者分裂点的选择十分关键，一旦一个合并或分裂被执行，就不能修正。

（2）不具有良好的可扩展性，因为合并或分裂的决定需要对大量的数据或簇进行计算。

基于层次的聚类算法的研究集中于凝聚层次聚类和迭代重定位方法的集成。其中典型的是 BIRCH 算法和 CURE 算法。BIRCH 算法首先用树结构对对象进行层次划分，然后采用其他的聚类算法对聚类结果进行求精；CRUE 算法采用固定数目的代表对象来表示每个簇，然后依据一个定义的分数向着聚类中心对它们进行收缩。

BIRCH（Balanced Lterative Reducing and Clustering using Hierarchies）算法是一种聚类算法，它可以在任何给定的内存下运行，主要用于处理超大规模的数据集。它是一个基于距离的层次聚类，综合了层次凝聚和迭代的重定位方法，首先用自顶向上的层次算法，然后用迭代的重定位来改进结果。算法的过程是将待分类的数据插到一棵树中，并且原始数据都在叶子节点上。它的主要思想是通过扫描数据库，建立一个初始存放于内存中的聚类特征树，然后对聚类特征树的叶节点加以聚类。算法的核心是聚类特征（Clustering Feature，简称 CF）和聚类特征树（Clustering Feature Tree，简称 CFT）。

BIRCH 算法具有如下几个特性。

（1）BIRCH 算法试图用有限的资源来生成最好的聚类结果，在给定的内存有限的情况下，将最小化 I/O 时间作为一个重要考虑因素。

（2）BIRCH 算法采用的技术是一种多阶段的聚类方法：只需对数据扫描一遍就能产生一个基本的聚类并有效地处理离群点，额外扫描可以起到改善聚类质量的作用。

（3）BIRCH 算法是一种增量的聚类方法，它对每一个数据点的聚类的决策不是基于全局的数据点，而是基于已处理过的数据点。

（4）BIRCH 算法比较适合球形的簇，如果簇不是球形的，则聚簇的效果将受到一定的影响。

绝大多数聚类算法要么擅长处理球形和相似大小的聚类，要么在存在孤立点时变得比较脆弱。而 CURE（Clustering Using REpresentatives）算法则解决了偏好球形和相似大小的问题，在处理孤立点上也更加健壮。

7.3 自组织映射网络

7.3.1 概述

自组织映射网络（Self-Organizing Maps，SOM）是 Kohonen 于 1981 年提出的非监督式学习方式，它通过模仿人脑神经元对信息的处理方式，进行自身训练，自动化对输入的模式进行聚类。自组织映射网络有两层，即输入层和输出层（也称作竞争层）。输入层和输出层通过权值向量连接。输入层的输入一般为高维向量，输出层的神经元通常放置在二维网格中，输出层邻居的神经元也通过权值连接，如图 7-4 所示。

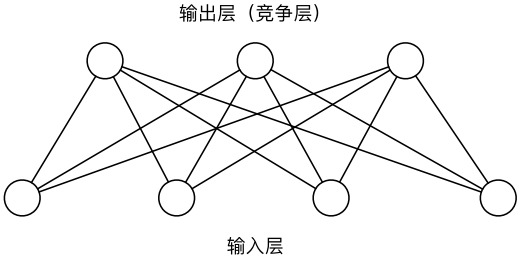


图 7-4 简单的自组织映射网络示例

SOM 则是基于模型的聚类方式，它的结构具有如下特征。

（1）网络的上层是输出节点（假设 m），二维排列成一个节点矩阵。

（2）如果输入向量为 n 个元素，则输入节点有 n 个节点。

（3）输出节点的所有输入节点都有连接权，输出节点的二维平面也可能具有本地连接。

（4）自组织映射网络的功能是通过自组织方法，用大量样本训练数据来调整网络的权值，使得最终网络输出能够反映样本数据的分布。

7.3.2 训练算法

自组织映射网络采用的是竞争学习规则方式对模型进行训练，可以自动发现输入数据之间的相似性，整个训练步骤如下。

① 初始化网络。随机对输入层和竞争层之间的权值给予初始值。

② 确定输入向量。将向量$\boldsymbol{x}=(x_1,x_2,\ldots,x_n)^{\mathrm{T}}$作为输入层的输入。

③ 计算竞争层的权值向量与输入向量之间的距离。在竞争层，计算神经元权值向量和输入向量的欧氏距离或者其他距离。其中，竞争层的第j个神经元与输入向量之间的距离为：

$$d=\sqrt{\sum_{i=1}^{n}(\boldsymbol{x}_i-w_{ij})^2}$$

上述公式中，w_{ij}表示输入层的i神经元和竞争层的j神经元之间的权值。

④ 选择加权向量最小距离的神经元。选择输入向量与权值向量之间最小距离的神经元，这个神经元则表示竞争获胜的神经元，表示为j^*，并给出相邻神经元的集合。

⑤ 调整权值。对于竞争胜利的神经元，其权值的变化量可参考如下公式，其中$h(j,j^*)$表示邻域函数，η表示当前时刻的学习速率：

$$\Delta w_{ij}=\eta h(j,j^*)(\boldsymbol{x}_i-w_{ij})$$

⑥ 不断迭代上述过程，直至模型收敛或达到最大迭代次数。

7.3.3　实例：利用自组织映射网络划分城市群

假定已知某 7 个城市的地理位置，并用经纬度表示，如表 7-1 所示，利用自组织映射网络将表 7-1 中的 7 个城市划分为三个城市群。

表 7-1　7 个城市的经纬度位置示例

城市代号	经　　度	纬　　度
c_1	E116°28′	N39°54′
c_2	E117°11′	N39°09′
1	E115°28′	N38°52′
c_4	E121°29′	N31°14′
c_5	E120°37′	N31°18′
c_6	E106°32′	N29°32′
c_7	E104°05′	N30°39′

由于输入的城市信息特征为地理位置，而地理位置通过经度和纬度两个维度进行表示，因此输入层的神经元为 7，代表每一个城市，维度为 2，特征表示为(x,y)，x表示经度，y表示纬度。由于需要将 7 个城市划分为三个城市群，因此竞争层神经元数量为 3，特征维度为 2，也采用(x,y)表示，分别代表一个城市群，如图 7-5 所示。

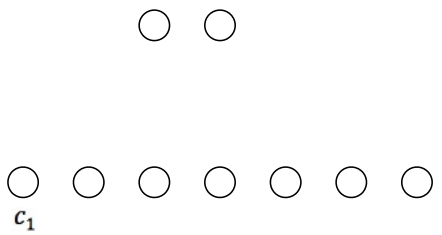


图 7-5　7 个城市划分为三个城市群的示例

基于图 7-5，因此基于自组织映射网络的计算思路如下。

1. 随机初始化三个输出节点$a_1(x_1, y_1)$、$a_2(x_2, y_2)$、$a_3\ (x_3, y_3)$。

2. 从城市c_1开始，分别对三个随机化初始的输出节点a_1、a_2、a_3进行距离计算，计算方式既可以采用基于经纬度的物理距离计算，也可以采用简易的欧氏距离计算。通过计算找到距城市c_1距离最短的a_k，作为竞争胜利的节点，并调整相应权值。这里的权值是指a_k的(x_k, y_k)的值，使它朝着离c_1更近的距离靠近，例如$\Delta w = (\eta(x_1 - x_k), \eta(y_1 - y_k))$。

3. 同理，依次对其他城市c_2、c_3、c_4、c_5、c_6、c_7采用与第二步相同的方式，寻找胜利节点，并调整权值，直到模型稳定。

模型的稳定除迭代次数外，与学习速率也有关系，一般情况下，随着迭代次数的增加，学习速率的值逐渐减小。例如，初始值 0.8，每次迭代乘以系数 0.95 等。上述在输入层的神经元确定时，也不同于前馈型神经网络的定义方式，虽然每一个城市的特征表示方式都一样，但是每一个城市都被作为一个神经元。原因在于如果按照前馈型神经网络的定义方式，则需要已知每个城市的城市群，则变成了分类问题，而非聚类问题。

通过上述过程的迭代计算，最终城市c_1、c_2、c_3为一个城市群；c_3、c_4为一个城市群；c_6、c_7为一个城市群。基于自组织映射网络的计算法方式，并不需要过于复杂的理论体系，这种无监督学习通过自适应的方式获得了问题的解。

7.3.4　优劣势分析

自组织映射网络算法可以进行有效的自适应分类，整体而言，自组织映射网络产生的结果非常容易理解且整个逻辑过程并不复杂，易于实现，但它仍然存在以下几方面问题。

（1）学习速率的选择使有必要在模型训练的速度和稳定性之间寻找平衡点。

（2）神经元的初始权值向量如果与实际输入向量的权值相差太大，则它很难在竞争中获胜，因此很可能成为无用的神经元。

（3）计算复杂度较高，对相似度的衡量方式比较敏感。这也是无监督学习的弊端，虽然无须训练集，但是要想达到收敛，就需要付出计算成本的代价，并且不太适合数据集较少的模型。

目前来看，自组织映射网络有比较宽阔的技术场景，例如，在自然语言处理、文档聚类、文件检索、自动查询、图像分割、数据挖掘、模糊分区、条件行动关联等场景中发挥其作用，在业务领域可以应用于医学诊断、数据压缩、生源分离、环境模型等。

7.4　其他自组织竞争型神经网络

7.4.1　自适应共振理论

1976 年，美国加州大学学者 G.A.Carpenter 提出了自适应共振理论（Adaptive Res-onance Theory，ART），他试图在人类心理学和认知活动间建立一个统一的数学理论，而 ART 是这一理论的核心部分。后来 G.A.Carpenter 和 S.Grossberg 提出了 ART 网络。经过多年的研究和不断地发展，ART 网络有三种形式。

（1）ART Ⅰ型处理双极性或二进制信号。

（2）ART Ⅱ型是 ART Ⅰ型的扩展形式，用于处理连续模拟信号。

（3）ART Ⅲ型是分级搜索模型，与前两个结构的功能兼容，并将两层神经网络扩展到任意多层神经网络。由于 ART Ⅲ型在神经元操作模型中纳入了生物神经元的电化学机制，因此具有较强的功能性和可扩展性。

ART 网络和算法在适应新的输入模式时具有更大的灵活性，同时避免了以前的模型的变化。ART 网络的核心点在于检查输入模式与存储模式的所有典型模式之间的匹配程度，以确定网络接收新输入时在预设参考阈值下的相似度。对于相似度超过阈值的所有模式类，选择与模式最相似的代表类，并调整与该类别相关联的权值，以便在模式之后的输入与模式匹配时可以获得相似度。如果相似度不超过阈值，则在网络中创建新的模式类，并创建与模式类相关联的权值，以表示和存储随后输入的模式及所有相似模式。

ART 自适应共振理论网络是竞争学习的重要代表之一。网络由比较层、识别层、识别层阈值和复位模块组成。ART 网络是减轻可塑性学习中的“可塑性—稳定性”的一种更好的方式。可塑性是指神经网络获取新知识的能力，稳定性则是指神经网络正在学习新知识时应该保留旧知识的记忆。这使得 ART 网络具有非常重要的优势：可以是增量学习或在线学习。

对于 ART I 型网络，基本的网络系统结构如图 7-6 所示。

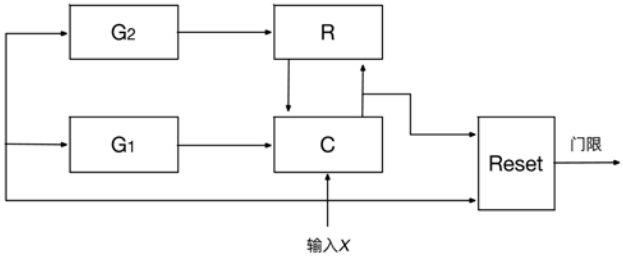


图 7-6 ART I 型网络的基本结构示例

在图 7-6 中，C 表示比较层，R 表示识别层，三种控制信号：R 为复位信号、G_1和G_2表示逻辑控制信号。

7.4.2 对偶传播神经网络

1987 年，美国学者罗伯特·赫尔特尼尔（Robert Hecht-nielson）提出了对偶传播神经网络（Counter Propagation Network，CPN），对偶传播神经网络能够存储二进制或者模拟值的模式对，这样的网络模型可以用于联想存储、统计分析、数据压缩等。与前馈型神经网络相比，执行同样的任务时，对偶传播神经网络训练周期更短，但是对偶传播神经网络的应用面比前馈型神经网络窄。

对偶传播神经网络的基本结构与具备三层结构的感知器有些相似，如图 7-7 所示。

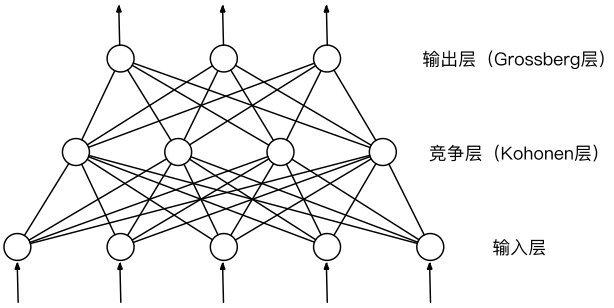


图 7-7 对偶传播神经网络的基本结构示例

与多层感知器相比，隐藏层在对偶传播神经网络的位置类似于竞争层，而输出层则为 Grossberg 层，分别采用无监督学习方式的竞争学习规则以及监督学习方式的 Widrow-Hoff 规则或 Grossberg 规则。

7.5　本章小结

自组织竞争型神经网络是基于竞争学习规则的学习网络模型，本章从自组织竞争型神经网络一般网络模型和工作原理入手，介绍自组织竞争型神经网络的理论体系。自组织竞争型神经网络作为一种常用的基于神经网络的聚类方法，本章详细介绍了机器学习领域中常见的几种聚类方式，包括系统聚类法、基于划分的聚类算法、基于密度的聚类算法和局域层次的聚类算法。

此外，本章重点介绍了自组织竞争型神经网络的典型代表——自组织映射网络，结合实例介绍了自组织映射网络的理论。除自组织映射网络外，自适应共振理论以及对偶传播神经网络也是自组织竞争型神经网络中比较典型的网络模型之一。

高阶篇

第 8 章

卷积神经网络

卷积神经网络（Convolutional Neural Network，CNN）是一种前馈型神经网络，其在大型图像处理方面有出色表现，目前已经被大范围使用到图像分类、定位等领域中。相比于其他神经网络结构，卷积神经网络需要的参数相对更少，使得其能够广泛应用。

8.1 概述

8.1.1 发展背景

卷积神经网络是目前深度学习技术领域中非常具有代表性的神经网络之一，在图像分析和处理领域取得了众多突破性的进展，在学术界常用的标准图像标注集 ImageNet 上，基于卷积神经网络取得了很多成就，包括图像特征提取分类、场景识别等。卷积神经网络相较于传统的图像处理算法的优点之一在于避免了对图像复杂的前期预处理过程，尤其是人工参与图像预处理过程，卷积神经网络可以直接输入原始图像进行一系列工作，至今已经广泛应用于各类图像相关的应用中。

从卷积神经网络的提出到目前的广泛应用，大致经历了理论萌芽阶段、实验发展阶段以及大规模应用和深入研究阶段。

（1）理论萌芽阶段。1962 年 Hubel 以及 Wiesel 通过生物学研究表明，从视网膜传递到大脑中的视觉信息是通过多层次的感受野（Receptive Field）激发完成的，并首先提出了感受野的概念。1980 年日本学者 Fukushima 在基于感受野的概念基础之上，提出了神经认知机（Neocognitron）。神经认知机是一个自组织的多层神经网络模型，每一层的响应都由上一层的局部感受野激发得到，对于模式的识别不受位置、较小形状变化以及尺度大小的影响。神经认知机可以理解为卷积神经网络的第一版，核心点在于将视觉系统模型化，并且不受视觉中物体的位置和大小等影响。

（2）实验发展阶段。1998 年计算机科学家 Yann LeCun 等提出的 LeNet-5 采用了基于梯度的反向传播算法对网络进行有监督的训练，Yann LeCun 在机器学习、计算机视觉等都有杰出贡献，被誉为卷积神经网络之父。LeNet-5 网络通过交替连接的卷积层和下采样层，将原始图像逐渐转换为一系列的特征图，并且将这些特征传递给全连接的神经网络，以根据图像的特征对图像进行分类。感受野是卷积神经网络的核心，卷积神经网络的卷积核则是感受野概念的结构表现。学术界对于卷积神经网络的关注，也正是开始于 LeNet-5 网络的提出，并成功应用于手写体识别。同时，卷积神经网络在语音识别、物体检测、人脸识别等应用领域的研究也逐渐开展起来。

（3）大规模应用和深入研究阶段。在 LeNet-5 网络之后，卷积神经网络一直处于实验发展阶段，直到 2012 年 AlexNet 网络的提出才奠定了卷积神经网络在深度学习应用中的地位，Krizhevsky 等提出的卷积神经网络 AlexNet 在 ImageNet 的训练集上取得了图像分类的冠军，使得卷积神经网络成为计算机视觉中的重点研究对象，并且不断深入。在 AlexNet 之后，不断有新的卷积神经网络提出，包括牛津大学的 VGG 网络、微软的 ResNet 网络、谷歌的 GoogLeNet 网络等，这些网络的提出使得卷积神经网络逐步开始走向商业化应用，几乎只要是存在图像的地方，就会有卷积神经网络的身影。

从目前的发展趋势而言，卷积神经网络将依然会持续发展，并且会产生适合各类应用场景的卷积神经网络，例如，面向视频理解的 3D 卷积神经网络等。值得说明的是，卷积神经网络不仅仅应用于图像相关的网络，还包括与图像相似的网络，例如，在围棋中分析棋盘等。

8.1.2　基本概念

卷积神经网络中有三个基本的概念：局部感受野（Local Receptive Fields）、共享权值（Shared Weights）、池化（Pooling）。

（1）局部感受野。对于一般的深度神经网络，往往会把图像的每一个像素点连接到全连接层的每一个神经元中，而卷积神经网络则是把每一个隐藏节点只连接到图像的某个局部区域，从而减少参数训练的数量。例如，一张 1024×720 的图像，使用 9×9 的感受野，则只需要 81 个权值参数。对于一般的视觉也是如此，当观看一张图像时，更多的时候关注的是局部。

（2）共享权值。在卷积神经网络的卷积层中，神经元对应的权值是相同的，由于权值相同，因此可以减少训练的参数量。共享的权值和偏置也被称作卷积核或滤波器。

（3）池化。由于待处理的图像往往都比较大，而在实际过程中，没有必要对原图进行分析，能够有效获得图像的特征才是最主要的，因此可以采用类似于图像压缩的思想，对图像进行卷积之后，通过一个下采样过程，来调整图像的大小。

8.1.3 基本网络结构

一个简单的卷积神经网络结构如图 8-1 所示。

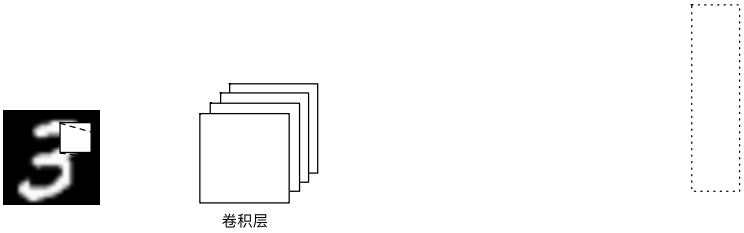


图 8-1 一个简单的卷积神经网络结构示例

基于图 8-1，一般的卷积神经网络包括卷积层、下采样层以及全连接层，其中卷积层和下采样层可以有更多层结构，并且卷积层和下采样层并不是一对一的关系，可以在多个卷积层的后面，连接一个下采样层，即 N 个卷积层相互之间叠加，然后再叠加一个下采样层，重复上述的结构 M 次。

从上述结构来看，与传统的全连接神经网络相比，卷积神经网络在输入层、卷积层、下采样层的结构上存在差异。理论上，全连接的前馈型神经网络具有丰富的特征表达能力，借助它可以很好地用于图形图像分析，但在实际使用过程中存在以下几方面问题。

（1）难以适应图像的变化性。对于图像而言，它拥有自己的局部不变性特征。例如，图像中有辆自行车，则该自行车出现在图像中的任意位置，甚至放大、缩小、旋转都不应该影响视觉对自行车的识别。而对于全连接的前馈型神经网络，则很难提取到这类特征，并且没有充分利用像素与像素之间的位置关系，这会导致识别率相对降低。若要提升识别率，则需要提升计算能力。

（2）计算量的复杂性。众所周知，图像是由像素点组成的，而图像一般有 3 个颜色通道，倘若对于一个 512×512 的图像采用全连接的前馈型神经网络进行分析，则输入层需要 786432（512×512×3）个输入单元，而全连接的前馈型神经网络层与层之间完全关联，再考虑到需要一定的神经网络深度，因此整个过程的计算量将会非常大。同时，全连接神经网络虽然深度越深表达能力越强，但是通过地图下降的方式训练全连接神经网络非常困难。若是一个浅层的全连接网络，则会导致整个全连接神经网络的训练效果大大降低，几乎不能使用。

卷积神经网络从局部连接、权值共享、下采样等方式有效解决了全连接遇到的问题。局部连接使得每一个神经元并不是和上一层的每一个神经元相连接，局部连接有效减少了训练过程中的参数量。权值共享可以使得一组连接共享权值，也减少了参数量，加快了训练速度。下采样使用池化的方式，减少每一个中的样本数量，并且提升了模型的鲁棒性。对于图像处理相关

的任务而言，卷积神经网络通过保留特征参数，以及减少不必要的参数，提升了训练速度，并保证了处理效果。

8.2　卷积

8.2.1　卷积的物理意义

在泛函分析中，卷积、叠积、摺积或旋积，是通过两个函数f和g生成第三个函数的一种数学算子，表征函数f与经过翻转和平移的g的乘积函数所围成的曲边梯形的面积，卷积公式如下：

$$\int_{-\infty}^{\infty} f(\tau)g(x-\tau)\mathrm{d}\tau$$

卷积的物理意义在于一个函数在另外一个函数上的加权叠加，不是简单的平移、翻转、反转等，而是对空间的变化，它是卷积神经网络的核心概念。

8.2.2　卷积的理解

以离散信号为例，已知：

$$x[0]=a_x, x[1]=b_x, x[2]=c_x$$

$$y[0]=a_y, y[1]=b_y, y[2]=c_y$$

两者的离散图如图 8-2 所示。

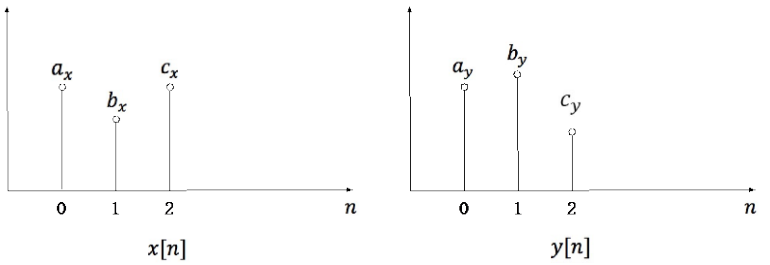


图 8-2　离散信号的离散图

可以通过$x[n]$×$y[n]$获得卷积的含义，对于$x[n]$和$y[n]$可以按照如下步骤计算。

1. 首先$x[n]$×$y[0]$，结果如图 8-3 所示。

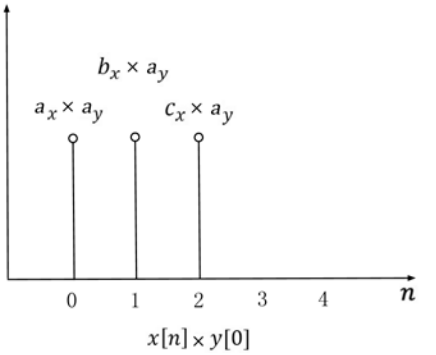


图 8-3　离散信号的$x[n] \times y[0]$结果示例

2 其次$x[n]$×$y[1]$，并向右移动 1 位，结果如图 8-4 所示。

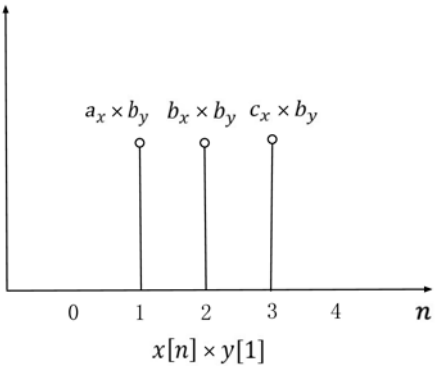


图 8-4　离散信号的$x[n]$×$y[1]$结果示例

3 然后将$x[n]$×$y[2]$，并向右移动两位，结果如图 8-5 所示。

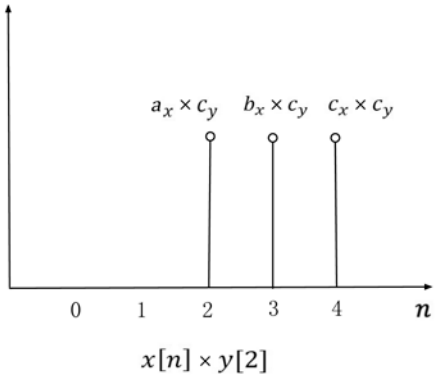


图 8-5　离散信号的$x[n]$×$y[2]$结果示例

最后将三者相加即得到了$x[n]$×$y[n]$的结果，如图 8-6 所示。

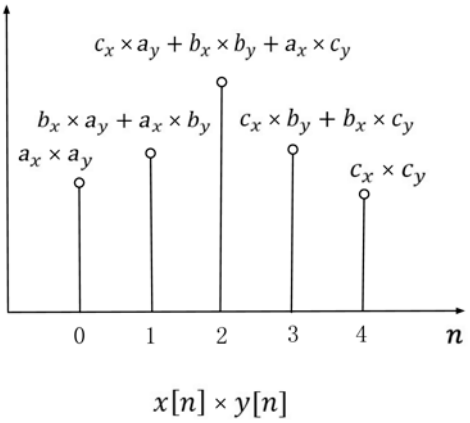


图 8-6　离散信号的$x[n]\times y[n]$结果示例

上述即为典型的加权叠加。

8.2.3　卷积的实例

假定某生物实验室正在培养 A 细菌，已知 A 细菌每天的繁殖率为 1%，即假设当前有 1 万个细菌样本，第二天新增 100 个，则按照这个速度可以计算每天拥有的细菌总数如表 8-1 所示。

表 8-1　每天拥有的细菌总数示例

初始样本数	第一天拥有样本数	第二天拥有样本数	第三天拥有样本数	第四天拥有样本数	第五天拥有样本数
+10000	$10000\times(1+0.01)^1$	$10000\times(1+0.01)^2$	$10000\times(1+0.01)^3$	$10000\times(1+0.01)^4$	$10000\times(1+0.01)^5$

到了第五天，实验室拥有 A 细菌的个数为$1000\times(1+0.01)^5$。

同时，假设隔壁实验室每天都向该生物实验室送来 10000 个 A 细菌，则该生物实验室每天拥有的细菌数变化情况如表 8-2 所示。

表 8-2　新增细菌之后，每天细菌的总数示例

初始样本数	第一天拥有样本数	第二天拥有样本数	第三天拥有样本数	第四天拥有样本数	第五天拥有样本数
+10000	$10000\times(1+0.01)^1$	$10000\times(1+0.01)^2$	$10000\times(1+0.01)^3$	$10000\times(1+0.01)^4$	$10000\times(1+0.01)^5$
	+10000	$10000\times(1+0.01)^1$	$10000\times(1+0.01)^2$	$10000\times(1+0.01)^3$	$10000\times(1+0.01)^4$
		+10000	$10000\times(1+0.01)^1$	$10000\times(1+0.01)^2$	$10000\times(1+0.01)^3$

续表

初始样本数	第一天拥有样本数	第二天拥有样本数	第三天拥有样本数	第四天拥有样本数	第五天拥有样本数
			+10000	10000 $\times(1+0.01)^1$	10000 $\times(1+0.01)^2$
				+10000	10000 $\times(1+0.01)^1$
					+10000

这样到第五天的时候，该生物实验室拥有的 A 细菌总数为：

$$\mathrm{SUM}=10000\times(1+0.01)^5+10000\times(1+0.01)^4+10000\times(1+0.01)^3+10000\times(1+0.01)^2+10000\times(1+0.01)^1+10000\times(1+0.01)^0$$

用公式可简化为：

$$\mathrm{SUM}=\sum_{i=0}^{5}f(i)\times g(5-i)$$

其中，$f(i)$可以理解为隔壁实验室的赠送函数，为常数 10000，$g(5-i)=(1+0.01)^{5-i}$表示细菌的繁殖函数，因此，生物实验室最终得到的 A 细菌总数即为赠送函数与细菌繁殖函数的卷积。

上述过程考虑的是非连续的情况，在上述基础之上，如果再给定两个条件：

（1）在 0~t的这段时间内，细菌不是隔天繁殖，而是时时刻刻都在繁殖，且每一刻的繁殖率都是 1%，则繁殖函数为$g(t-\tau)=(1+0.01)^{t-\tau}$。

（2）隔壁实验室也不是每天都向该生物实验室送新的 A 细菌，而是时时刻刻都在送，则赠送函数为$f(\tau)$。

因此在t时刻时，生物实验室拥有的 A 细菌总数为：$\int_0^t f(\tau)g(t-\tau)\mathrm{d}\tau=\int_0^t f(\tau)\times(1+0.01)^{t-\tau}\,\mathrm{d}\tau$。

8.3 卷积核

8.3.1 卷积核的含义

在图形图像处理中，有一种基本的工具：滤波器。对于图像的滤波则是利用一个滤波器对

图像进行卷积操作，滤波器也被称作卷积核。滤波器是一个矩阵。例如，有如下任意的卷积核：

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{matrix}$$

常见的卷积核如下所示。

（1）对图像无任何影响的卷积核：

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

（2）对图像进行锐化的滤波器：

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{matrix}$$

（3）浮雕滤波器：

$$\begin{matrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{matrix}$$

（4）均值模糊滤波器：

$$\begin{matrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{matrix}$$

均值模糊是对像素点周围的像素值进行均值化处理，将上下左右及当前像素点分作 5 份，然后进行平均，每份占 0.2，即对当前像素点周围的点进行均值化处理。

（5）高斯模糊滤波器。

均值模糊是一种简单的模糊处理方式，但是会显示模糊不够平滑，而高斯模糊可以很好地处理，因此高速模糊经常用于图像的降噪处理上，尤其是在边缘检测之前，进行高斯模糊，可以移除细节带来的影响。在常见的图像处理软件中均能够见到，例如 Photoshop、Paint.NET 等。

（6）一维高斯函数：

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{x^2}{2\sigma^2}}$$

（7）二维高斯函数：

$$G(x) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{x^2+y^2}{2\sigma^2}}$$

8.3.2 卷积操作

滤波器是图像处理的基本方式，对于一个图像，在卷积核的基础上，进行 2D 卷积操作。2D 卷积则是对图像中的每一个像素点，将它的相邻像素组成的矩阵与滤波器矩阵的对应元素相乘，然后将相乘后的结果进行相加，得到该像素的最终值，如图 8-7 所示。

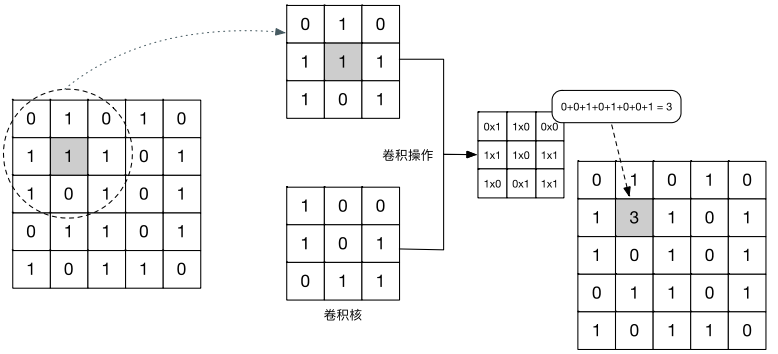


图 8-7 卷积操作的示例

对像素中的每一个像素点均按照上述过程进行卷积操作，即完成对图像的处理。在遇到图像顶部或底部的边缘像素时，可以按照如下方式进行处理。

（1）忽略边缘像素。将边缘像素直接不纳入到卷积的操作中。

（2）填充边缘像素。对边缘进行扩展，并保证卷积操作能够包含到边缘像素，扩展的边缘像素值可以填充为 0，也可以采用中心点的像素值代替填充像素值，或者使用中心点附近的平均像素值代替。

8.3.3 卷积核的特征

滤波器也有自己的特征：

（1）当滤波器矩阵中的值相加为 0 甚至更小时，被滤波器处理之后的图像相对会比原始图像暗，值越小越暗。

（2）当滤波器矩阵中的值相加为 1 时，被滤波器处理之后的图像与原始图像的亮度相比几乎一致。

（3）当滤波器矩阵中的值相加大于 1 时，被滤波器处理之后的图像相对会比原始图像的亮度更亮。

8.4　卷积神经网络中各层工作原理

8.4.1　卷积层

卷积层是卷积神经网络不同于其他神经网络的关键之处，卷积层有效地考虑了图像的变化情况，例如图像的平移、放大、缩小、旋转等，如图 8-8 所示。

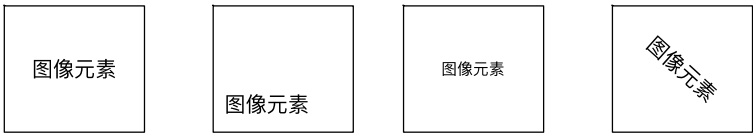


图 8-8　图像的平移、缩小、旋转示例

卷积层不再采用全连接的方式，而是采用部分连接，如图 8-9 所示。

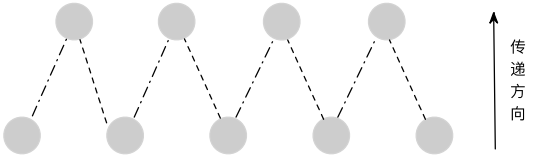


图 8-9　卷积层的部分连接示例

图 8-9 中下层的神经元不再是每一个神经元与上层的每一个神经元相连，而是与相关的关联，除此之外，图 8-9 中相同样式的边对应的权值也是相同的，这是卷积层的一个特性：权值共享。

8.4.2　下采样层

卷积层通过非全连接的方式显著减少了神经元的连接，从而减少了计算量，但是神经元的数量并没有显著减少，对于后续计算的维度依然比较高，并且容易出现过拟合问题。为解决此问题，在卷积神经网络的卷积层之后，会有一个池化层（Pooling），用于池化操作，也被称作子采样层（Subsampling），子采样层可以大大降低特征的维度，减少计算量，同时可以避免过拟合问题。下采样层的操作过程示例如图 8-10 所示。

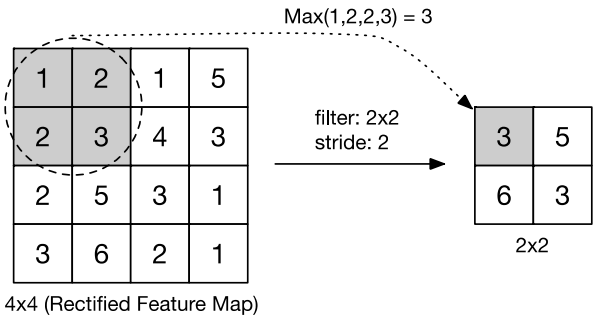


图 8-10　下采样的操作过程示例

以图 8-10 为例，对于一个 1024×1024 的特征图进行 2×2 的池化操作，即每 4 个元素中取最大的一个元素作为输出，最终得到的是 512×512 的特征图，规模减少了四分之一。

8.4.3　Softmax 层

Softmax 函数是概率论中常见的归一化函数，它能够将K维的向量$\boldsymbol{x}$映射到另外一个K维向量$\boldsymbol{p}(\boldsymbol{x})$中，并使得新的$K$维向量的每一个元素取值在(0,1)区间，且所有$K$维向量之和为 1。公式如下所示：

$$p(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{K} e^{x_j}} \quad for\ i = 1,2,\dots,K$$

例如，输入三维向量[5.0,2.0,3.0]，通过 Softmax 函数得到的三维向量为[0.844, 0.042, 0.114]。计算过程如图 8-11 所示。

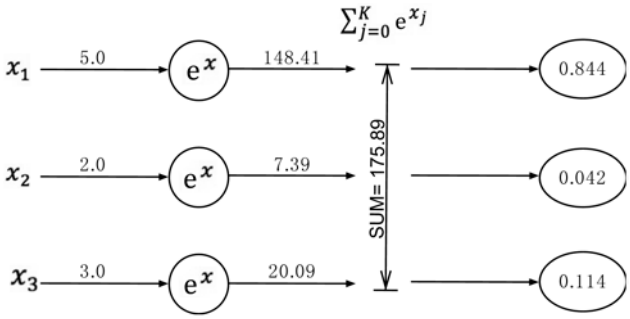


图 8-11　Softmax 函数的计算示例

作为一种归一化函数，与其他的归一化方法相比，Softmax 函数有着其独特的作用，尤其在多分类问题中，均有广泛应用，例如，朴素贝叶斯分类器以及神经网络中。它的特点是，在对向量的归一化处理过程中，尽可能凸显较大值的权值，抑制较小值的影响，因此在分类应用

中可以更加凸显分类权值较高的类别。

8.5　卷积神经网络的逆向过程

一般情况下，卷积神经网络不会涉及逆向过程，但是目前在比较热门的研究领域，例如，图像自动生成中会涉及卷积的逆向过程，其中主要包括上采样过程、反激活过程和反卷积过程。

（1）上采样过程。池化过程本身是不可逆的，因此反池化过程是不能够还原图像本身内容的，但是可以在池化过程中，记录其位置。例如，采用 Max Pooling 时，可以记录 Max Pooling 的值来源位置，在反池化过程中使用，将该位置的值填入，其余的位置置为 0 即可，如图 8-12 所示。

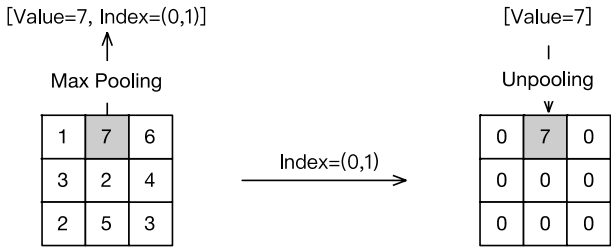


图 8-12　反池化过程示例

图 8-12 中的左侧部分是 Max Pooling 过程，右侧部分是 Unpooling 过程，池化过程的块大小为 3×3，通过最大池化过程之后，获得的最大值为 7，并记录其索引位置为(0,1)，即最大池化将一个 3×3 的块变为了一个 1×1 的块，且块的值为 7，来源于原始块的(0,1)位置。反池化过程恰好相反，是将一个 1×1 的块变为一个 3×3 的块，此时则需要借助在池化过程中记录的原始块的位置，将 1×1 块的内容填充到原始记录的位置，其余全部置为 0，如图 8-13 所示，即得到图 8-12 的右侧部分。

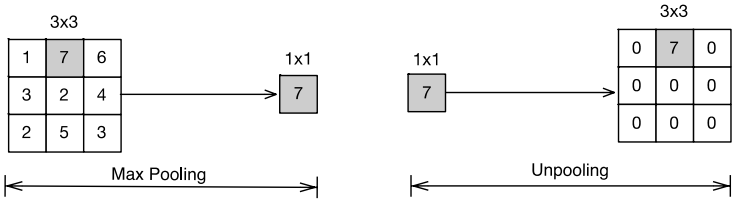


图 8-13　池化过程和反池化过程对比示例

（2）反激活过程。在激活过程中采用的是 ReLU 函数及其衍生函数，对于反激活的过程也是需要保证所有值都为正数，因此反激活过程依然可以采用 ReLU 函数及其衍生函数。

（3）反卷积过程。反卷积的特征是可视化，反卷积主要用于可视化一个已经训练好的卷积网络模型，这个过程不会涉及模型的训练。反卷积计算过程与卷积过程类似，只不过采用的滤波器有所改变，是对卷积过程中的滤波器进行了转置。

8.6 常见卷积神经网络结构

8.6.1 LeNet-5

LeNet-5 是一种典型的卷积神经网络，它主要用于手写字和印刷字识别，它由 Yann LeCun 等人在 1998 年发表的论文 *Gradient-Based Learning Applied to Document Recognition* 中呈现，其结构如图 8-14 所示。

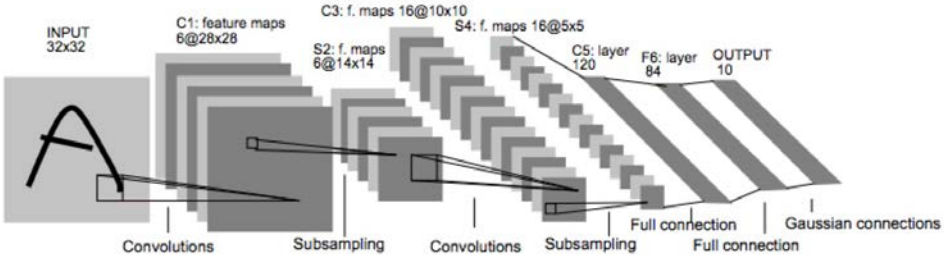


图 8-14 LeNet-5 的结构（来自 Yann LeCun 等著论文 *Gradient-Based Learning Applied to Document Recognition*）

在图 8-14 中，INPUT 表示一个输入层，定义的输入图像大小为 32×32，卷积核大小为 5×5，卷积核的种类个数为 6 个，C1 表示经过卷积操作之后的特征图，卷积输出的特征图为 28×28，计算方式为$28 = 32 - 5 + 1$。

在 LeNet-5 中定义了 6 个不同的卷积核种类，因此神经元的数量为$28\times28\times6$，对于每一个卷积核可以训练的参数为$(5\times5+1)$（其中 1 为偏置项），因此一共可以训练的参数为$(5\times5+1)\times6$，从而总的连接数为$(5\times5+1)\times6\times$（28×28）。

S2 层为下采样层，输入为 28×28 的矩阵，采样的区域大小为 2×2，采样的方式为采用四个输入值相加，然后乘以一个可训练的参数，再加上偏置项，通过 Sigmoid 函数进行激活。由于是在 28×28 的矩阵中，进行区域为 2×2 的采样，因此输出的特征图大小为 14×14。采样的种类数量为 6 个，则神经元的数量为 14×14×6，可以训练的参数为 2×6，因此总的连接数为(2×2+1)×6×14×14，通过和 C1 层的比较可知，S2 层的输出特征图大小大小只有 C1 层的 1/4。

C3 层属于卷积层，输入为 S2 层中的 6 个或者多个特征图的组合，卷积核大小为 5×5，卷

积核的种类数量为 16，输入的特征图大小为 10×10。C3 层中的每个特征图都是 S2 层中的 6 个或多个特征图的组合，表示本层的特征图是上一层提取到的特征图的不同组合。

训练方式可以采用 C3 的前 6 个特征图和 S2 层的 3 个相邻的特征图子集作为输入；之后 6 个特征图采用 S2 层的 4 个相邻的特征图子集作为输入，后面 3 个则采用不相邻的四个特征图子集作为输入，最后一个则采用 S2 层的所有特征图作为输入，因此可以训练的参数数量为：

$$6\times(3\times25+1)+6\times(4\times25+1)+3\times(4\times25+1)+1\times(6*25+1)=1516$$

因此，连接数量为 1516×(10×10)=151600。

S4 是一个下采样层，输入是 10×10 的矩阵，采样区域为 2×2，采样方式同 S2 层一样是四个输入相加，然后乘以一个可以训练的参数，再加上偏置，激活函数采用 Sigmoid 函数。

根据输入矩阵大小和采样区域，可知输出的特征图大小为 5×5，采样种类的数量设定为 16，因此神经元的数量为 5×5×16=400，可以训练的参数为 32（2×16），总的连接数为 2000（16×（2×2+1）×5×5）。从这个过程中也可以看出，S4 层输出的特征图从是 C3 层输出特征图大小的 1/4。

C5 是最后一个卷积层，C5 层的输入是 S4 层的输出特征图，并且与 S4 层是全连接关系，不再是 S2 层与 C3 层的组合关系。定义卷积核的大小为 5×5，卷积核的种类数量为 120，输出的特征图大小为 1×1（5-5+1），因此可以训练的参数为 48120〔（16×（5×5+1）×120）〕。

F6 层为全连接层，有 84 个单元，输入是 C5 层输出的 120 维向量，计算方式与经典的神经网络相同，采用输入向量与权值向量之间的点积，再加上偏置，可以训练的参数数量为 84×（120+1）=10164，最终激活函数采用 Sigmoid 函数。

上述即为影响了卷积神经网络发展的 LetNet-5 卷积神经网络，通过上述介绍不难发现，LetNet-5 中的 5 是指卷积层与下采样层的数量之和。

8.6.2　AlexNet

AlexNet 于 2012 年出现在 ImageNet 的图像分类比赛中，并取得了 2012 年的冠军，从此卷积神经网络开始受到人们的强烈关注。AlexNet 是深度卷积神经网络研究热潮的开端，也是研究热点从传统视觉方法到卷积神经网络的标志性网络结构。

AlexNet 模型一共有八层，包含五个卷积层和三个全连接层，对于每一个卷积层，均包含了 ReLU 激活函数和局部响应归一化处理，接着进行了下采样操作。AlexNet 的模型结构如图 8-15 所示。

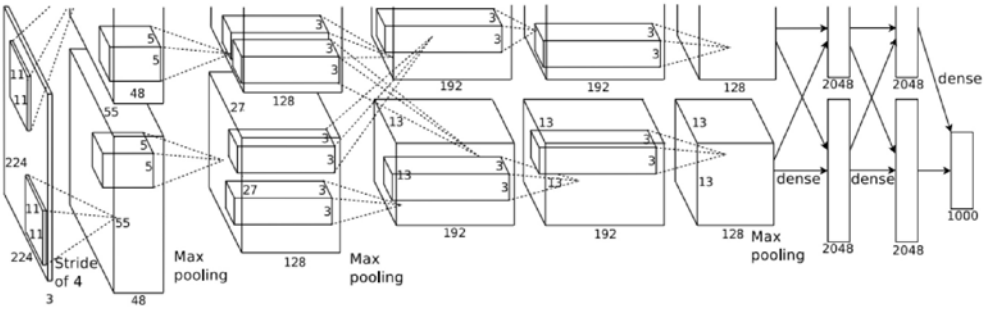


图 8-15　AlexNet 的模型结构（来自 Alex Krizhevsky 等著论文 *ImageNet Classification with Deep Convolutional Neural Networks*）

（1）从图 8-15 中可以发现，输入的图像大小为 224×224，但是考虑到图像是由 RGB 组成的三通道，因此输入图像的规则是 224×224×3。（在 AlexNet 模型的实际处理过程中会通过预处理，图像的输入规则是 227×227×3）

（2）在 AlexNet 模型中，一共有 96 个 11×11 的卷积核进行特征提取，考虑到图像为 RGB 图像，则存在三个通道，因此这 96 个滤波器实际使用过程中也是 11×11×3，即原始图像是彩色图像，那么提取的特征也是具有彩色的特质。特征图大小的计算方式采用如下公式：

$$\text{特征图大小} = \frac{(\text{图像大小} - \text{卷积核大小})}{\text{stride}} + 1$$

其中stride表示步长，决定了卷积核滑动的大小。因此第一层卷积层生成的特征图大小为(227-11)/4+1=55，因此第一层卷积层得到的特征图一共是 96 个，每一个特征图的大小是 55×55，且带 RGB 三通道。

（3）AlexNet 模型使用 ReLU 函数作为激活函数，使得特征图中特征值取值范围在合理范围。

（4）AlexNet 模型中使用到了局部区域归一化处理的方式。比如内核是 3×3 的矩阵，则该过程是对 3×3 区域的数据进行处理，通过下采样处理可以得到(55-3)/2+1=27，得到 96 个 27×27 的特征图，这 96 个特征图将会作为第二次卷积层操作的输入。

第二层卷积层的输入是第一层卷积层输出的 96×27×27 的特征图，采用的是 256 个 5×5 大小的卷积核，利用卷积核对输入的特征图进行处理，处理方式与第一层卷积层的处理方式略有不同。过滤器是对 96 个特征图中的某几个特征图中相应的区域乘以相应的权值，然后加上偏置之后对所得区域进行卷积，第二层卷积层计算完毕之后，得到的是 256 个 13×13（(27-3)/2+1=13）的特征图。

后续的操作类似，第三层卷积层没有采用下采样操作，得到的是 384 个 13×13 的新特征图；第四层依然没有进行下采样操作，因此得到的依然是 384 个 13×13 的特征图，第五层得到的是

256 个 6×6 的特征图。

对于第六层的全连接层，使用 4096 个神经元，并对第五层输出的 256 个 6×6 的特征图进行全连接；第七层的全连接层与上一层全连接层类似；第八层全连接层采用的是 1000 个神经元，对第七层的 4096 个神经元进行全连接，然后通过高斯滤波器，得到各项预测的可能性。

在训练模型过程中，会通过实际值与期望值进行误差计算，求出残差，并通过链式求导法则，将残差通过求解偏导数逐步向上传递，在神经网络的各个层不断修改权值和偏置，和一般的神经网络思想类似。

值得说明的是，局部响应值归一化（Local Response Normalization，LRN）是借助神经生物学中的侧抑制思想，即当某个神经元被激活时，抑制相邻的其他神经元。局部响应值归一化的目的是实现局部的抑制，ReLU 也是一种“侧抑制”方法。这样做的意义在于有利于增强模型的泛化能力。LRN 通过模仿神经生物学中的侧抑制思想，对局部的神经元实现了竞争机制，使得响应比较大的值相对能够更扩大。

8.7　应用场景与效果评估

8.7.1　场景 1：图像分类

1. 图像分类经典模型

图像分类是根据图像中呈现的信息中所反映的不同特征，将不同类别的图像区分开来的方法。图像分类利用卷积神经网络对图像进行各项特征分析，并计算出图像中每个像元或区域属于各种类别的概率，最终确定图像的分类。深度卷积神经网络为更细粒度的图像分类带来了更好的效果，这种细粒度更体现出计算机对图像的理解，例如，基于图像美感的分类。大量基于卷积神经网络的思想提出的新网络结构，促进了图像分类领域的快速发展。

在图像分类的模型中，比较经典的是 VGG16 卷积神经网络模型，VGG 模型的架构是由 Simonyan 和 Zisserman 在 2014 年发表的论文 *Very Deep Convolutional Networks for Large Scale ImageRecognition* 中提出的。VGG16 模型的结构非常简单，由 13 个卷积层和 3 个全连接层组成，总共 16 层的网络结构，并通过 Softmax 进行图像分类。VGG16 模型能够以较高的图像分类准确度识别日常生活中的 1000 种不同种类的物体。

基于卷积神经网络的图像分类模型成为了图像处理的基础模型，不仅 VGG16 模型，还包括 VGG19、ResNet50、InceptionV3、Xception 等经典的网络模型，都成为图像处理的预训练模型。

2. 图像分类的效果评估方式

在图像分类中，需要对图像分类的结果进行评估，每个算法的性能通常使用多类别混淆矩阵及其派生统计来评估。在获得混淆矩阵后，可以评估分类器的整体性能和单个分类器的性能，以全面评估分类模型的性能。

对于多分类问题的整体分类精度，通常使用整体分类精度进行评估。整体分类精度仅考虑对角线方向正确对齐的像素数，而 Kappa 系数则考虑到对角线以外的各种泄漏和错误分类像素。Kappa 系数可用于评估分类模型的整体准确性。当 Kappa 系数的值大于 0.80 时，表示分类数据和测试数据的一致性较高，即分类精度较高。

整体分类精度的评估并不能完全反映单一分类器的性能。一般来说，可以根据混淆矩阵得到每个分类器的识别率以及准确度，绘制 ROC 曲线和 PR 曲线，以评估表分类器的准确性。

平均精度（AP）最初用于评估信息的 IR，其平均值为不同召回率的正确率。直观上，AP 是 PR 曲线下的区域，平均值是检索平均值。平均精度、平均值意味着平均所有类别（每个类被视为次要任务）。现在图像分类论文基本上是以 mAP 为标准。

对于大型数据集，例如 ImageNet 共有 1000 万张图像，大量样本的计算量非常大。数据集的分类任务评估使用总体错误率，累积分类误差时累积值为 1。该指标是整体精度的补充。前 1 个错误和前 5 个错误用于评估和排名每个模型的结果。最后，使用 top-5 错误作为图像分类的评估指标。

8.7.2 场景 2：目标检测

1. 目标检测实例

目标检测是基于目标的几何和统计特征的图像区域识别，首先需要识别到目标，其次是识别目标在图像的位置，目标可以为人脸、动物、汽车等，目标检测的效果如图 8-16 所示。

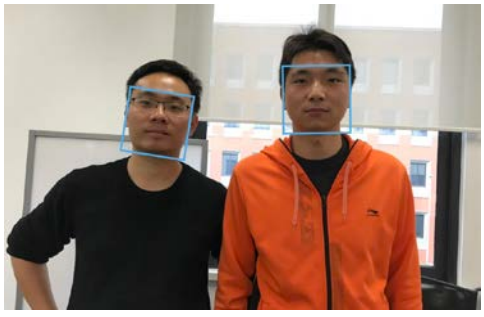


图 8-16 对图像中的人脸进行检测的示例

目标检测在现实生活中非常有用，例如，人脸检测可以用于在照相过程中调节焦距，根据识别的目标进行计数；物体检测时可以进行智能裁剪，甚至对目标进行提取，以使得其与其他技术混合进行二次处理。目标检测与图像分类不同，目标检测不仅可以对图像像素进行分类，还可以定位目标位置。

2. 目标检测的效果评估

一般而言，可以通过实例分割的准确度、图像分割的精度、过分割率以及欠分割率等指标对图像分割的结果进行评估。评估的重要参考则是 Ground Truth 图像，Ground Truth 图像可以理解为期望的分割结果图像。分割精度主要体现在图像分割的真实区域面积占 Ground Truth 图像中期望区域面积的百分比。过分割率表示在 Ground Truth 图像中期望区域之外被分割的像素面积占比。欠分割率表示在 Ground Truth 图像中还未达到期望区域的像素面积占比。

像素级图像分割的评估也可以转化为分类模型的评估，ISPR（International Society for Photogrammetry and Remote Sensing）提出了一种基于累积混淆矩阵的经典分类精度评估方法，比如通过计算分割精度的精度，以评估分割性能等。经典的 VOC2007（PASCAL Visual Object Classes Challenge 2007）数据集中使用的评估指标是平均分割精度，即计算所有类别的分割精度的算术平均值。

对于大型数据集 ImageNet，计算每个类别的测试结果的准确率（Recall）。每个目标类别的最终评估指标是平均精度（AP），即 PR 曲线的积分值。单一目标类别检测 AP 最高胜出，检测目标类别获得最高数量的团队，以赢得目标检测挑战。

8.7.3　场景 3：实例分割

1. 实例分割

自 2012 年 Alex Krizhevsky、Geoff Hinton 和 Ilya Sutskever 赢得了 ImageNet 的冠军之后，卷积神经网络就成为了分割图像的利器。现在，卷积神经网络通过不断完善和改进，在某些视觉领域已经超过人类。

与实例分割相近的概念是像素级的语义分割，后者是将每一个像素划分到对应的类别，实现像素级的分类，而实例分割是在语义分割的基础上，不仅要在像素级上进行分类，还需要在具体的类别上划分出不同的示例。例如，图 8-17 通过像素的语义分割，识别狗在原图中的像素区域。

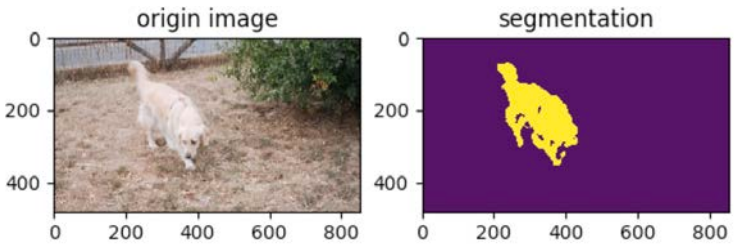


图 8-17　对图像中的狗进行实例分割的示例

目前绝大多数的实例分割都是基于卷积神经网络的，例如 R-CNN、Mask R-CNN、Fast R-CNN、Faster R-CNN 等。

2. 实例分割的效果评估

像素级图像分割的评估也可以转化为分类模型的评估。ISPR 提出了一种基于累积混淆矩阵的经典分类精度评估方法。首先，将原始图像分割为正面照片（TOP）瓷砖，然后对瓷砖中每个像素的类别进行计数，构建基于像素的混淆矩阵。最后，每个瓦片的混淆矩阵被累积得到累积混淆矩阵。一旦获得了累积混淆矩阵，就可以计算精度的精度、召回、F1 分数等，以评估分割性能。

PASCAL Visual Object Classes Challenge 2007（VOC2007）数据集中使用的评估指标是平均分割精度，即所有类别的分割精度的算术平均值。每个类的分割精度的程度是正确除以地面实数的像素数（实际精度精度）的像素数。

8.8 MAXOUT NETWORKS

在深度卷积神经网络中，存在三种常见的网络结构变化，分别是：Maxout Networks、Network In Network、Global Average Pooling，其中，Maxout Networks 是改进中比较典型的结构。

Maxout 可以视为深度神经网络的激活函数层，一般的前向传播过程中，隐藏节点的输出公式如下：$h_i(x) = \text{sigmoid}(x^{\mathrm{T}}W_i + b_i)$。

一般的神经网络传递过程如图 8-18 所示。

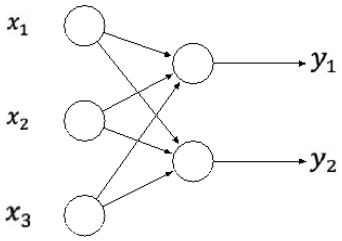


图 8-18　一般的神经网络传递过程

第i层有两个输入，对于第$i+1$层，则有：

$$z = x_1 \times w_1 + x_2 \times w_2 + b$$

$$y = f(z)$$

其中f为激活函数，而对于添加 Maxout 层之后的计算公式则为：

$$h_i(x) = \max_{j \in [1,k]} z_{ij}$$

其中z_{ij}为：

$$z_{ij} = x^{\mathrm{T}} W_{ij} + b_{ij} \ (W \in \mathbb{R}^{d \times m \times k})$$

W是一个三维的数据，d表示输入层节点的数量，m表示隐藏层节点的数量，k表示在隐藏层之后新增的k个节点，Maxout 则是在k个节点中，选择最大的一个作为输出。

同样，在图 8-18 添加 Maxout 层之后，选定$k = 5$，结果如图 8-19 所示。

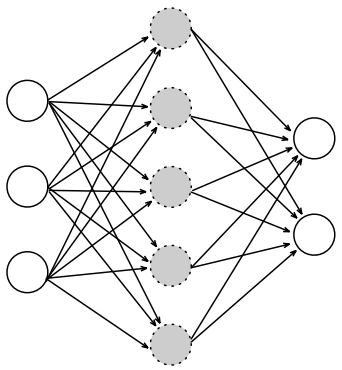


图 8-19　添加 Maxout 层神经网络结果示例

相当于在原始的第i层和第$i+1$层之间添加了新的一层，对于 Maxout 的输出公式则为：

$$z_1 = w_1 \times x + b_1$$

$$z_2 = w_2 \times x + b_2$$

$$z_3 = w_3 \times x + b_3$$

$$z_4 = w_4 \times x + b_4$$

$$z_5 = w_5 \times x + b_5$$

$$\text{out} = \max(z_1, z_2, z_3, z_4, z_5)$$

Maxout 可以被看作一个激活函数，而它又不同于传统的激活函数。Maxout 激活函数并不是一个固定的函数表达式，不同于 Sigmoid、ReLU、Tanh 等激活函数，它是一个固定的函数方程。Maxout 有较强的拟合能力，它是一个分段线性函数，而任意的凸函数都可以由分段线性函数进行拟合，因此它几乎可以拟合任意的凸函数。在前馈型神经网络中，Maxout 输出取最大值。 在卷积神经网络中，Maxout 特征图可以通过多个特征图获得最有价值的特征。

8.9 本章小结

本章由浅入深地介绍了卷积神经网络。首先介绍了卷积神经网络的发展背景，卷积神经网络的发展经历了三个重要阶段，目前正处于被广泛使用和深入研究的阶段。接着介绍了卷积神经网络中的重要概念卷积、卷积核等，不仅介绍了卷积神经网络中的重要层次结构，如卷积层、下采样层、Softmax 层，还介绍了卷积神经网络的逆向过程。

本章通过介绍常见的卷积神经网络，使得读者能够理解经典的卷积神经网络，并通过应用实例介绍了图像分类、目标检测以及图像分割，以帮助读者深入理解卷积神经网络的工作原理和价值。

第 9 章

循环神经网络

人脑可以记住以前发生过的事情，并作为后续预测行为的基础。传统的神经网络很难做到这一点，这是传统神经网络潜在的弊端。例如，基于影片中的每个时间点对事件类型进行分类，传统的神经网络很难求解这类带有时间序列的问题。循环神经网络（Recurrent Neural Networks，RNN）是一个包含时间循环结构的网络，它允许信息被持久化。RNN 本质上是一个特殊结构的神经网络，正是基于这样的结构特性，才使得它可以处理具备时序相关信息的能力。

9.1 概述

传统神经网络能够实现对物体的分类和标注，当处理有前后依赖信息的预测问题时，就显得力不从心。该问题的本质是由传统神经网络的结构决定的，因此要处理联想预测的问题时，就需要设计新的网络结构来实现。这种网络的输出不仅依赖当前的输入，还需要结合前一时刻或后一时刻的输入作为参考，因此该网络属于反馈型神经网络类型。循环神经网络是反馈型神经网络中处理时序相关反馈的典型代表，循环神经网络目前在应用中代表了大多数反馈型神经网络。

循环代表着神经网络中的反馈，传统的反馈型神经网络由 Jordan 于 1986 年提出，1990 年 Elman 针对语音问题提出了另外一种形式略有不同的反馈型神经网络，引入从隐藏层传递信息的可能性，即具有局部记忆单元和局部反馈连接的网络，也就是前文介绍的 Elman 网络。循环的引入同样带来了长期依赖的难题，即经过多层次的网络传播之后，出现梯度消失或梯度爆炸问题。直到 1997 年 Hochreiter 和 Schmidhuber 引入被称为 LSTM 的神经元之后，问题才得以解决。如今世面上基于循环神经网络的应用，大都是基于这样的神经元或者变种组成的网络。

9.2 一般循环神经网络

9.2.1 概述

对于传统的经典神经网络，简化其结构如图 9-1 所示。

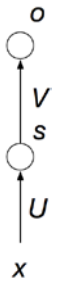


图 9-1 经典神经网络结构图示例

整个结构分为三层；输入层、隐藏层、输出层。其中，$\boldsymbol{x}$表示输入层，是一系列特征，用向量的形式进行表示；$\boldsymbol{s}$为隐藏层的输出，采用向量的形式，隐藏层可以有多个神经元，神经元的个数即s的维度；$\boldsymbol{o}$表示为输出层的值，依然采用向量的形式，具体可以为物体的分类等。一般分类的个数就是$\boldsymbol{o}$的维度。$\boldsymbol{U}$是输入$\boldsymbol{x}$特征与隐藏层神经元全连接的权值矩阵；$\boldsymbol{V}$则是隐藏层与输出层全连接的权值矩阵。$\boldsymbol{s}$的输出由权值矩阵$\boldsymbol{U}$以及输入$\boldsymbol{x}$来决定，$\boldsymbol{o}$的输出由权值矩阵$\boldsymbol{V}$和隐藏层输出$\boldsymbol{s}$决定。

循环神经网络结构如图 9-2 所示。

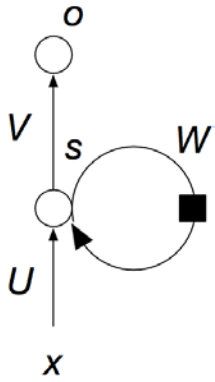


图 9-2 循环神经网络结构示例

通过对比图 9-1 与图 9-2 可以发现，两者在整体上非常相似，都是由输入层、隐藏层和输出层构成，最大的区别点在于循环神经网络隐藏层多了一个自身到自身的环形连接，即s的值

不再仅依赖$\boldsymbol{U}$和$\boldsymbol{x}$，还依赖新的权值矩阵$\boldsymbol{W}$以及上一次$\boldsymbol{s}$的输出。其中，$\boldsymbol{W}$表示上一次隐藏层的输出到这一次隐藏层输入的权值矩阵。由于隐藏层多了一个自身到自身的循环输入，因此该层被称作循环层。该网络结构即循环神经网络，图 9-2 也是最简单的反馈型神经网络形式。循环神经网络有多种类型，按照循环的方向分类，可以分为单向循环神经网络和双向循环神经网络两种；根据循环的深度分类，则可以分为循环神经网络和深度循环神经网络两种。

9.2.2　单向循环神经网络

图 9-2 表示的循环神经网络结构即为单向循环神经网络，现对其展开之后的效果如图 9-3 所示。

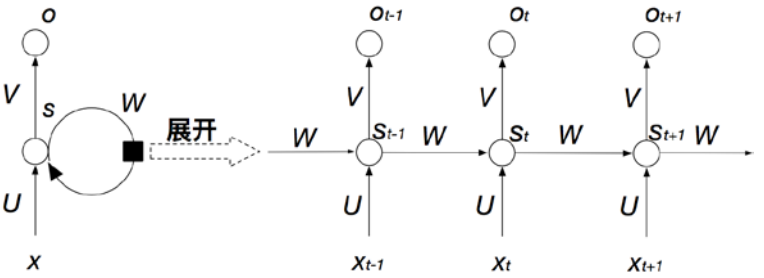


图 9-3　单向循环神经网络结构展开示例

根据图 9-3 所示，可以理解为网络的输入通过时间进行向后传递。当前隐藏层的向量输出$\boldsymbol{s_t}$除了取决于当前输入层的输入向量$\boldsymbol{x_t}$外，还受到上一时刻隐藏层的输出向量$\boldsymbol{s_{t-1}}$的影响，因此，当前时刻隐藏层的输出信息包含了之前时刻的信息，表现出对之前信息记忆的能力。可以采用如下公式对单向循环神经网络进行表示：

$$\boldsymbol{o_t} = g(\boldsymbol{V}\boldsymbol{s}_t)$$

$$\boldsymbol{s_t} = f(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{W}\boldsymbol{s}_{t-1})$$

其中$\mathbf{o}_t$表示输出层的计算公式，输出层与隐藏层通过全连接进行，g为输出层的激活函数，$\boldsymbol{V}$为输出层的权值矩阵。$\mathbf{s}_t$表示隐藏层的计算公式，它接收两类输入，即当前时刻的输入层输入$\boldsymbol{x}_t$及上一时刻的隐藏层输出$\mathbf{s}_{t-1}$，U为输入层到隐藏层的权值矩阵，$\boldsymbol{W}$为上一时刻的值$\mathbf{s}_{t-1}$到作为这一次输入的权值矩阵，f为隐藏层的激活函数。值得说明的是，$\boldsymbol{U}$、$\boldsymbol{V}$、$\boldsymbol{W}$权值矩阵的值每次循环都是一份，因此循环神经网络的每次循环步骤中，这些参数都是共享的，这也是循环神经网络的结构特征之一。依据上述的递归定义，现对公式进行展开：

$$\begin{aligned} o_t = g(\boldsymbol{V}\boldsymbol{s}_t) &= Vf(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{W}\boldsymbol{s}_{t-1}) = \boldsymbol{V}f(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{W}f(\boldsymbol{U}\boldsymbol{x}_{t-1} + \boldsymbol{W}\boldsymbol{s}_{t-2})) \\ &= \boldsymbol{V}f(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{W}f(\boldsymbol{U}\boldsymbol{x}_{t-1} + \boldsymbol{W}f(\boldsymbol{U}\boldsymbol{x}_{t-2} + \cdots))) = \ldots \end{aligned}$$

从上述公式中可以看出，$\boldsymbol{o}_t$为$\boldsymbol{x}_t$、$\boldsymbol{x}_{t-1}$、$\boldsymbol{x}_{t-2}$、……的函数，因此当前时刻的输出会受到历史输入的综合影响。

9.2.3 双向循环神经网络

在日常的信息推断中，当前信息不单单依赖之前的内容，也有可能会依赖后续的内容。比如针对英语的完形填空，为了获得要填写的单词，往往需要获取待填写位置之前和之后的单词，统一理解后再去推断应该填写的单词。完形填空的场景则是典型的信息双向依赖问题，该类问题非单向循环神经网络能够处理。单向循环神经网络仅仅能够从之前输入的信息进行推断，而不能够将后续的信息纳入参考，因而会影响最终结果的准确性。

针对完形填空，双向循环神经网络（Bi-directional Recurrent Neural Network，BRNN）依据前后文的信息中进行记忆推断，得出正确的填写单词。双向循环神经网络是由 Schuster 和 Paliwal 于 1997 年提出的，其主要思想是训练一个分别向前和向后的两个循环神经网络，表示完整的上下文信息，两个循环神经网络都对应同一个输出层，因此可以简单理解为是两个循环神经网络的叠加，输出结果是基于两个循环神经网络的隐藏层的输出状态计算获得。将双向循环神经网络按照时间序列进行结构展开，如图 9-4 所示。

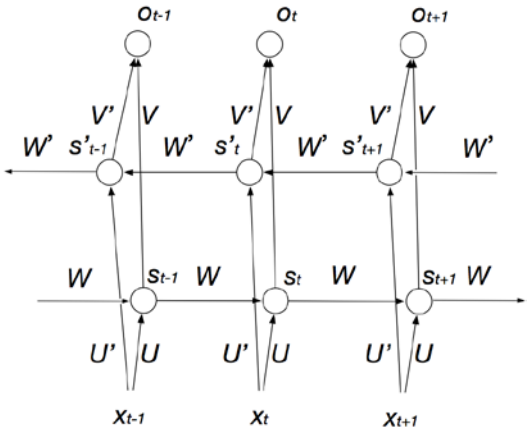


图 9−4　双向循环神经网络结构展开示意图

从图 9-4 可以看出，网络的隐藏层要保留两部分信息：一部分为从前到后的正向信息传递**s**，另一部分为从后到前的反向信息传递$\boldsymbol{s}'$，最终的输出$\boldsymbol{o}$则综合了这两部分信息。按单向循环神经网络的推算方式，不难得出双向循环神经网络的每一层计算公式：

$$\boldsymbol{o}_t = g(\boldsymbol{V}\boldsymbol{s}_t + \boldsymbol{V}'\boldsymbol{s}'_t)$$

$$\boldsymbol{s}_t = f(\boldsymbol{W}\boldsymbol{s}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t)$$

$$\boldsymbol{s}_t' = f(\boldsymbol{W}'\boldsymbol{s}_{t-1}' + \boldsymbol{U}'\boldsymbol{x}_t)$$

从$o_t = g(\boldsymbol{V}\boldsymbol{s}_t + \boldsymbol{V}'\boldsymbol{s}_t')$可以看出，某一时刻的输出$o_t$受到$s_t$和$s_t'$影响。从$\boldsymbol{s}_t = f(\boldsymbol{W}\boldsymbol{s}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t)$、$\boldsymbol{s}_t' = f(\boldsymbol{W}'\boldsymbol{s}_{t-1}' + \boldsymbol{U}'\boldsymbol{x}_t)$的递归定义来看，分别包含了正向和反向两部分的信息。值得说明的是，从上述三个公式来看，正向计算和反向计算是不共享权值的，也就是说，输入层对于隐藏层的权值矩阵U、U'，隐藏层对于输出层的权值矩阵$\boldsymbol{V}$、$\boldsymbol{V}'$，隐藏层对于隐藏层内部的双向权值矩阵$\boldsymbol{W}$、$\boldsymbol{W}'$，这 6 个独特的权值在每一时刻都被复用，而正反向两两之间的值是不相同的。同时，从图 9-4 中也可以看到，向前和向后的隐藏层之间是没有连接的，这样的展开图是非循环的。

9.2.4　深度循环神经网络

上述介绍的单向循环神经网络和双向循环神经网络仅仅包含一个隐藏层，而在实际应用中，为了增加表达能力，往往引入多个隐藏层，即深度循环神经网络。一个简单的多层循环神经网络结构如图 9-5 所示。

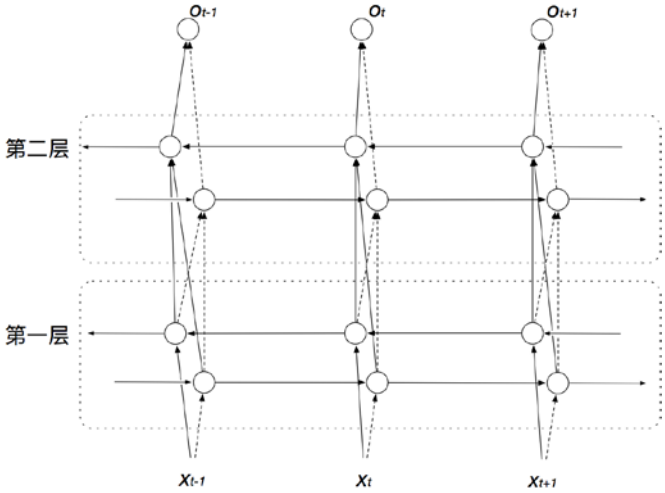


图 9-5　多层循环神经网络结构展开示意图

同样可以得到深度循环神经网络的计算公式：

$$\boldsymbol{o}_t = g(\boldsymbol{V}^{(i)}\boldsymbol{s}_t^{(i)} + \boldsymbol{V}'^{(i)}\boldsymbol{s}_t'^{(i)})$$

$$\boldsymbol{s}_t^{(i)} = f(\boldsymbol{W}^{(i)}\boldsymbol{s}_{t-1} + \boldsymbol{U}^{(i)}\boldsymbol{s}_t^{(i-1)})$$

$$\boldsymbol{s}_t'^{(i)} = f(\boldsymbol{W}'^{(i)}\boldsymbol{s}_{t+1}' + \boldsymbol{U}'^{(i)}\boldsymbol{s}_t'^{(i-1)})$$

…

$$\boldsymbol{s}_t^{(1)} = f(\boldsymbol{W}^{(1)}\boldsymbol{s}_{t-1} + \boldsymbol{U}^{(1)}\boldsymbol{x}_t)$$

$$\boldsymbol{s}_t'^{(1)} = f(\boldsymbol{W}'^{(1)}\boldsymbol{s}'_{t+1} + \boldsymbol{U}'^{(1)}\boldsymbol{x}_t)$$

上述公式中，上标i表示当前的层数。通过以上各公式可以看出，最终的输出依赖两个维度的计算，横向上内部前后信息的叠加，即按照时间的计算；纵向上是每一时刻的输入信息在逐层之间的传递，即按照空间结构的计算。

9.3 训练算法：BPTT 算法

在循环神经网络中的反向传播算法被称之为 BPTT（Back Propagation Through Time），主要针对循环层的训练。由于循环层包含两部分的输入，即当前时刻的输入与上一时刻循环层的输出，因此循环神经网络的训练算法相比较传统的神经网络训练算法多了一步向前计算的过程，但是其基本原理与 BP 算法一致，整体上包含以下三个步骤：

1. 前向计算每个神经元的输出值；
2. 反向计算每个神经元误差项的值；
3. 计算每个权值的梯度。

并采用梯度下降算法更新权值。

9.3.1 前向计算

这一步主要是计算$\boldsymbol{s}_t$的输出，可以通过如下公式进行计算：

$$\boldsymbol{s}_t = f(\boldsymbol{W}\boldsymbol{s}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t)$$

其中$\boldsymbol{s}_t$、$\boldsymbol{x}_t$、$\boldsymbol{s}_{t-1}$为某一时刻的向量，下标表示某一时刻t，$\boldsymbol{W}$、$\boldsymbol{U}$为矩阵，f表示激活函数。假设输入x是m维的，输出状态为n维，则U矩阵的大小为$n\times m$，$\boldsymbol{W}$矩阵的大小为$n\times n$，将上式展开如下：

$$\begin{bmatrix} s_1^t \\ s_2^t \\ \vdots \\ s_n^t \end{bmatrix} = f\left(\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix} \begin{bmatrix} s_1^{t-1} \\ s_2^{t-1} \\ \vdots \\ s_n^{t-1} \end{bmatrix} + \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1m} \\ u_{21} & u_{22} & \cdots & u_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nm} \end{bmatrix} \begin{bmatrix} x_1^t \\ x_2^t \\ \vdots \\ x_m^t \end{bmatrix} \right)$$

$\boldsymbol{w_{ji}}$表示$t-1$时刻隐藏层神经元i与t时刻神经元j的连接权值，$\boldsymbol{u_{ji}}$表示上一层神经元i与下一层神经元j的连接权值。

9.3.2　误差项计算

计算循环神经网络的某一时刻t的误差，可以按照两个方向进行传播：一个按照空间结构传递到上一层，这一部分的误差主要受到权值矩阵$\boldsymbol{U}$影响；另一个按照时间方向上传递到上一个时刻，这一部分的误差主要受到权值矩阵$\boldsymbol{W}$影响，计算公式参考如下：

$$\delta_k^{\mathrm{T}} = \delta_t^{\mathrm{T}} \prod_{i=k}^{t-1} \boldsymbol{W} \cdot \mathrm{diag}[f'(\mathrm{net}_i)]$$

上式公式表示误差项按照时间反向传播的计算方法，δ_k为任意时刻k的误差项，net_i表示i时刻神经元的加权输入，拆开表示为：$\boldsymbol{W}s_{i-1} + \boldsymbol{U}x_i$。由此看来，某一时刻的误差计算依赖之后时刻的误差值以及权值矩阵的值，体现了误差按照时间反向传播的思想。

另一方面，循环层误差按照结构的反向传播到上一层，与之前段落介绍的全连接层的反向传播完全一致。给出计算公式如下：

$$(\delta_t^{l-1})^T = (\delta_t^l)^T U \cdot \mathrm{diag}[f'^{l-1}(\mathrm{net}_t^{l-1})]$$

δ_t^{l-1} 表示$l-1$层t时刻误差，net_t^{l-1}表示$l-1$层神经元的加权输入，当前层的误差受到它后面一层δ_t^l的误差及U的影响，体现了误差按照结构反向传播的思想。

9.3.3　权值梯度计算

至此依据前向计算和误差计算，已经可以获取到任意时刻的隐藏层状态输出s_t及任意时刻的误差项δ_k和δ_t^{l-1}，前者为时间反向传播的误差，后者为结构反向传播的误差。依据上述结果可以按照如下公式得出权值矩阵$\boldsymbol{W}$在t时刻的梯度：

$$\nabla w_t E = \begin{bmatrix} \delta_1^t s_1^{t-1} & \delta_1^t s_2^{t-1} & \dots & \delta_1^t s_n^{t-1} \\ \delta_2^t s_1^{t-1} & \delta_2^t s_2^{t-1} & \dots & \delta_2^t s_n^{t-1} \\ \vdots & \vdots & \vdots & \vdots \\ \delta_n^t s_1^{t-1} & \delta_n^t s_2^{t-1} & \dots & \delta_n^t s_n^{t-1} \end{bmatrix}$$

δ_i^t表示t时刻误差项向量的第i个分量，s_i^{t-1}表示$t-1$时刻循环层第i个神经元的状态输出值。依据δ_i^t以及s_i^{t-1}可以计算任意时刻$\boldsymbol{W}$的梯度，最终梯度为各个时刻的梯度之和：

$$\nabla wE = \sum_{i=1}^{t} \nabla w_i E$$

进而可以通过$\boldsymbol{W} = \boldsymbol{W} - \eta \nabla wE$进行时间反向的梯度更新计算。与权值矩阵$\boldsymbol{W}$计算类似，不难得出权值矩阵$\boldsymbol{U}$的计算方法：

$$\nabla u_t E = \begin{bmatrix} \delta_1^t x_1^t & \delta_1^t x_2^t & \cdots & \delta_1^t x_m^t \\ \delta_2^t x_1^t & \delta_2^t x_2^t & \cdots & \delta_2^t x_m^t \\ \vdots & \vdots & \vdots & \vdots \\ \delta_n^t x_1^t & \delta_n^t x_2^t & \cdots & \delta_n^t x_m^t \end{bmatrix}$$

同理，最终梯度也是各个时刻的梯度之和：$\nabla uE = \sum_{i=1}^{t} \nabla u_i E$，再通过$\boldsymbol{U} = \boldsymbol{U} - \eta \nabla uE$进行结构上反向的梯度更新计算。

9.3.4 梯度爆炸与梯度消失问题

上述训练过程可以较好地训练模型的参数，但是循环神经网络在实际应用中，与其他网络一样，会遇到不稳定的梯度问题，即梯度消失和梯度爆炸问题，从而导致训练时的梯度不能够在较长序列中传递下去，从而无法获取长距离的影响。根据公式$\delta_k^T = \delta_t^T \prod_{i=k}^{t-1} W \cdot \mathrm{diag}[f'(\mathrm{net}_i)]$可以得到：

$$||\delta_k^{\mathrm{T}}|| \leqslant ||\delta_t^{\mathrm{T}}|| \prod_{i=k}^{t-1} ||W|| \cdot ||\mathrm{diag}[f'(\mathrm{net}_i)]|| \leqslant ||\delta_k^{\mathrm{T}}|| (\beta_w \beta_f)^{t-k}$$

上述公式中，β表示矩阵模的上界，整体为一个指数函数，$t-k$表示从t时刻到k时刻的距离，这个相差越大，误差项的值抖动就越大，从而进一步影响到权值的更新幅度，产生大幅抖动或者无更新，即表现为梯度消失或梯度爆炸问题。这取决于β的值是大于 1 或者小于 1。

对于梯度爆炸问题相对容易解决，一般情况下梯度爆炸的时候计算过程会收到类似 NaN 错误，因此可以设置一个梯度阈值，当梯度超过这个阈值的时候直接截取即可。而梯度消失问题相对不容易解决，当前主要采用以下三种方式解决梯度消失问题：

（1）对权值矩阵设置合理的初始化值，尽量避免每个神经元不要取极大值或者极小值，以躲避梯度消失的区域。

（2）使用不容易产生梯度消失的激活函数，如用 ReLU 函数代替Sigmoid和函数 Tanh函数。

（3）设计特殊结构的循环神经网络，如长短时记忆网络（LSTM）或 Gated Recurrent Unit（GRU）。

9.4 长短时记忆网络

9.4.1 背景

长短时记忆网络（Long Short Term Memory Network，LSTM）是一种特殊结构的循环神经

网络，可以学习长期依赖信息，成功解决了传统的循环神经网络的缺陷，并成为当下比较流行的循环神经网络结构，在语音识别、图像描述、自然语言处理等领域中能达到较好的应用效果。

传统的循环神经网络引入了时序反馈机制，可以解决信息关联问题。但是 BPTT 算法引发的梯度爆炸和梯度消失，使得网络随着输入序列的增长，抖动变得更为剧烈，导致无法学习。为了解决此问题，Hochreater 和 Schmidhuber 于 1997 年提出了长短时记忆网络，即长短时记忆网络结构，引入恒定误差传送带或 CEC 单元解决 BPTT 中的梯度爆炸和梯度消失问题。经过 Alex Graves 的改良和推广，在解决很多实际问题上取得了巨大成功，并得到了更为广泛的应用。

9.4.2　核心思想

在 9.3.4 节中已经通过公式给出产生梯度爆炸与梯度消失的原因。梯度爆炸相对较为容易处理，针对梯度消失问题，长短时记忆网络的核心思想在于：传统的循环神经网络的循环层只有一个状态s，它对短期的输入非常敏感，这时就不免想到通过内部增加一个状态来记忆长期信息以解决问题，差异对比如图 9-6 所示。

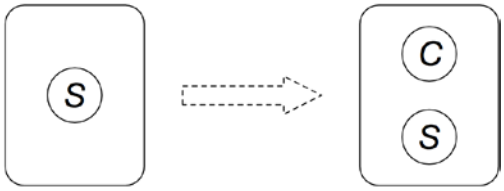


图 9-6　经典 RNN 与 LSTM 核心对比图

在图 9-6 中，左图为传统循环神经网络的神经元结构，右图为长短时记忆网络的神经元结构，新引入的c被称之为单元状态（Cell State），其主要职责就是做长期信息记忆之用。现将图 9-6 中的右图按照时间展开，得到如图 9-7 所示效果。

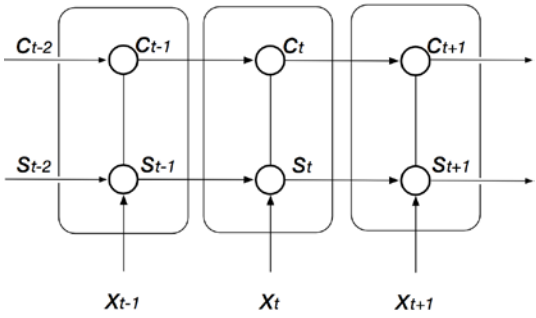


图 9-7　长短时记忆网络结构展开图

从图 9-7 可以看出，在某一时刻t，长短时记忆网络的神经元输入由三部分组成：当前网

络的输入$\boldsymbol{x}_t$、上一时刻的输出$\boldsymbol{s}_{t-1}$以及上一时刻的单元状态$\boldsymbol{c}_{t-1}$，神经元的输出为当前时刻的输出$\boldsymbol{s}_t$，当前时刻的单元状态为$\boldsymbol{c}_t$，$\boldsymbol{c}$与$\boldsymbol{x}$、$\boldsymbol{s}$一样，都是向量。从图 9-7 中可以看出长短时记忆网络的神经元内部$\boldsymbol{s}$的输入处理方式在形式上与传统的循环神经网络几乎一致。多出来的单元状态$\boldsymbol{c}$与内部的$\boldsymbol{s}$产生联系，并接收上次输入，同时将本次状态传递给下一个神经元。

长短时记忆网络的核心则是基于单元状态c进行控制。核心思路即引入三个控制开关：第一个开关负责控制是否继续保持长期状态$\boldsymbol{c}$，在图 9-7 中即为$\boldsymbol{c}_{t-1}$到$\boldsymbol{c}$的输入部分；第二个开关负责是否将当前的输入信息输出到长期状态$\boldsymbol{c}$中，即$\boldsymbol{s}$到$\boldsymbol{c}$的过程；第三个开关负责控制是否将当前的状态$\boldsymbol{c}$作为当前神经元的输出，即$\boldsymbol{c}$到$\boldsymbol{s}$的过程。三个开关的示意图如图 9-8 所示。

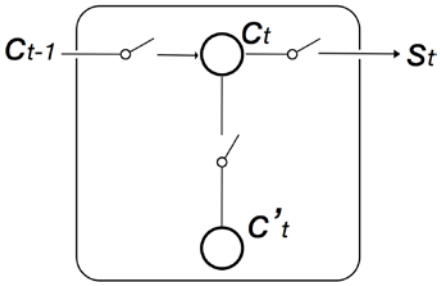


图 9-8　长短时记忆网络内部开关示意图

在图 9-8 中，$\boldsymbol{c}_{t-1}$表示上一时刻的长期状态，$\boldsymbol{c}_t$表示当前时刻的长期状态，$\boldsymbol{c'}_t$表示当前时刻的即时状态，$\boldsymbol{s}_t$为当前时刻的输出。

9.4.3　详细结构

对于传统的循环神经网络结构可以简化为如图 9-9 所示。

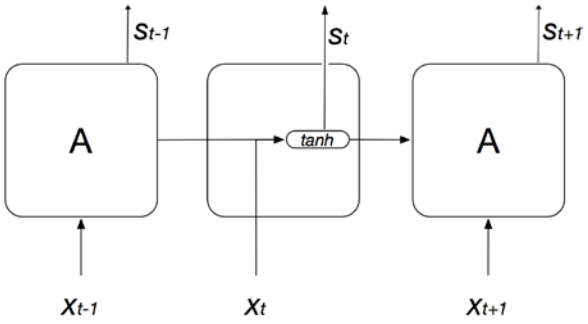


图 9-9　传统循环神经网络结构图

从图 9-9 中可以看出，其结构非常简单，每个神经元内部仅仅是一个单一的tanh层。长短时记忆网络神经元内部的详细结构图如图 9-10 所示。

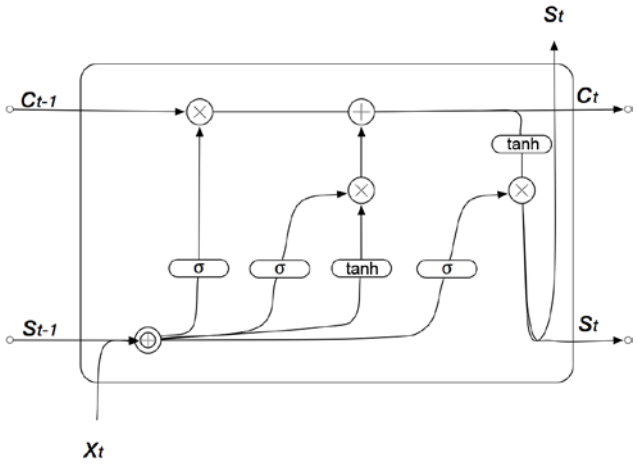


图 9-10　长短时记忆网络神经元内部详细结构图

在图 9-10 中，“⊕”表示向量按元素连接符号。“⊗”表示为按元素逐乘操作。“⊕”表示按元素逐加操作。“σ”、“tanh”表示神经网络层。

通过对比，可以发现长短时记忆网络神经元内部不再只是用一个单一的$tanh$层，而是引入了四个相互作用的层，即三个σ、一个$tanh$。差别最大也是最关键的结构，为贯穿$\mathbf{c}_{t-1}$到$\boldsymbol{c}_t$的水平线。该水平线就像一条信息的传送带，内部只对传入的向量信息做少量的线性操作，继续传递到下一层，很容易地完成了信息的前后传递，进而实现了长期记忆保留的功能。

如果仅依靠该水平线是无法直接完成信息有选择更新的，因此需要引入一种叫作门（Gate）的结构来实现对输入神经元的信息进行过滤。

长短时记忆网络中引入了三种门结构来保护和控制信息，分别为：遗忘门（Forget Gate Layer）、输入门（Input Gate Layer）和输出门（Output Gate Layer），这三种门的结构主要通过一个sigmoid的神经层和一个逐点相乘的操作来完成。

1. 遗忘门

首先长短时记忆网络要解决的是决定让哪些信息继续通过这个神经元，这个功能结构即遗忘门。它主要决定上一时刻的$\boldsymbol{s}_{t-1}$与$\boldsymbol{c}_{t-1}$状态是否保留到当前时刻的$\boldsymbol{c}_t$当中。具体是通过一个$Sigmoid$神经层和一个连接$\boldsymbol{c}_{t-1}$的逐点相乘计算来实现的，如图 9-11 所示。

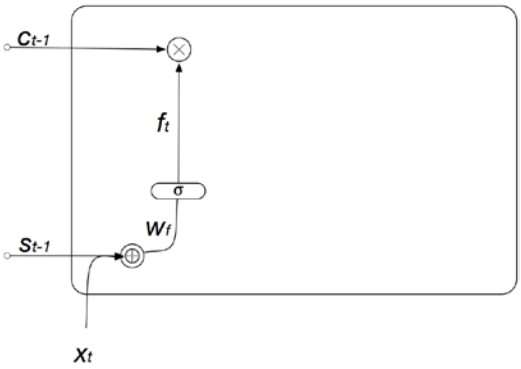


图 9-11 长短时记忆网络遗忘门结构图

它的输入为上一时刻的输出$\boldsymbol{s}_{t-1}$与当前时刻的输入$\boldsymbol{x}_t$，经过Sigmoid函数变换，得到内部当前时刻输出$\boldsymbol{f}_t$。具体公式表达如下：

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_f)$$

上述公式中，$\boldsymbol{W}_f$表示遗忘门的权值矩阵，$[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t]$表示两向量纵向连接操作，$b_f$表示输入的偏置项。

2. 输入门

输入门决定了当前时刻输入$\boldsymbol{x}_t$，有多少需要保存到当前单元状态$\boldsymbol{c}_t$。此部分功能的实现由两个层组成：一个sigmoid层来决定哪些输入被更新，一个tanh层生成一个向量作为备选更新信息，这两层的输出进行逐点相乘，从而对单元状态$\boldsymbol{c}_t$进行更新，如图 9-12 所示。

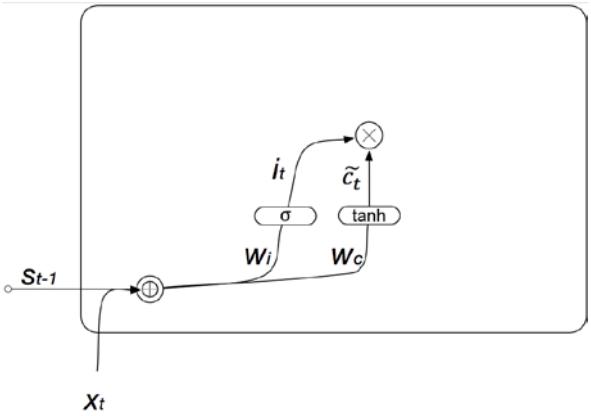


图 9-12 长短时记忆网络输入门结构图（a）

依据图 9-12，不难得出输出门的计算输出公式：

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_i)$$

$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}_c[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_c)$$

可以看出在计算$\boldsymbol{i}_t$和$\tilde{\boldsymbol{c}}_t$时，使用的权值矩阵和偏置项是不同的，因此在训练的过程中需要单独训练。

有了上面两个门结构后，则可以对单元状态$\boldsymbol{c}_t$进行更新操作，如图 9-13 所示。

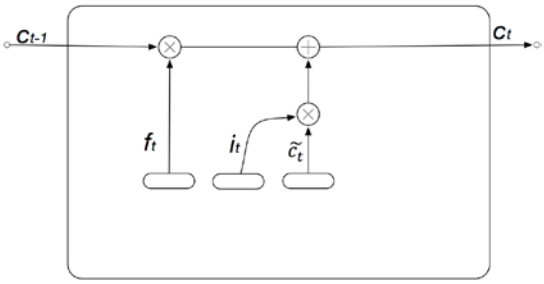


图 9-13 长短时记忆网络输入门结构图（b）

进而可以得出单元状态$\boldsymbol{c}_t$的计算公式：

$$\boldsymbol{c}_t = \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \tilde{\boldsymbol{c}}_t$$

为了方便输入，符号“∘”表示按元素逐乘计算。不难理解，$\boldsymbol{f}_t$的输出非，即 1。$\boldsymbol{f}_t \circ \boldsymbol{c}_{t-1}$就是通过元素相乘，把不想保留的信息进行忘却操作。$\boldsymbol{i}_t \circ \tilde{\boldsymbol{c}}_t$就是将当前的信息有选择地输入到当前单元状态。最后通过元素相加将两者的信息进行叠加更新为$\boldsymbol{c}_t$。

3. 输出门

完成信息的选择性记忆与更新后，接下来需要考虑如何将当前的信息进行输出，即引出输出门的设计，该门主要来控制单元状态$\boldsymbol{c}_t$有多少可以输出到长短时记忆网络的当前输出值$\boldsymbol{s}_t$中，如图 9-14 所示。

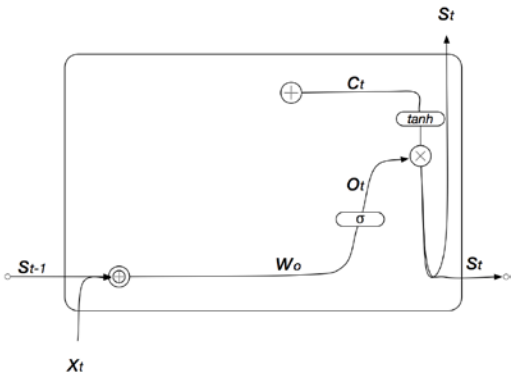


图 9-14　长短时记忆网络输出门结构图

根据图 9-14 可以看出，该单元的输出主要依赖当前的神经元状态c_t，不只是单纯依赖单元状态，还需要进行一次信息过滤的处理，即由引入的sigmoid层来完成。这一层将单元状态经过tanh层处理后的数据进行元素相乘操作，将得到的s_t有选择地输出到下一时刻和对外输出。具体计算公式如下：

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_o)$$

$$\boldsymbol{s}_t = \boldsymbol{o}_t \circ \tanh(c_t)$$

9.4.4 训练过程

前文完整分析了长短时记忆网络的详细结构，有一定的复杂程度。长短时记忆网络的训练也是非常重要的过程。长短时记忆网络作为循环神经网络，其训练算法采用的是反向传播算法，结合长短时记忆网络，其训练过程主要有三个步骤。

1. 通过前向计算获得每个神经元的输出值，这对长短时记忆网络来说，即$\boldsymbol{f}_t$、$\boldsymbol{i}_t$、$\boldsymbol{c}_t$、$\boldsymbol{o}_t$、$\boldsymbol{s}_t$向量的值。

2. 通过反向计算获得每个神经元的误差项值，与传统循环神经网络一样，误差项的计算涉及两个方向：一个是按照时间序列的反向传播，即从当前时刻t开始计算前一时刻$t-1$误差项的值；另一个是按照空间结构的反向传播，即从当前层l向上一层$l-1$计算误差项的值。

3. 在两个方向误差项值的基础上，计算每个权值矩阵的梯度值，并进行更新操作。

在具体分析之前，为了方便理解后面各个计算公式的含义，首先对部分公式进行前置定义，具体如下。

1. 激活函数及导数定义

门的激活函数及其导数定义：

$$\sigma(z) = y = \frac{1}{1+\mathrm{e}^{-z}}$$

$$\sigma'(z) = y(1-y)$$

输出激活函数tanh及其导数定义：

$$\tanh(z) = y = \frac{\mathrm{e}^{z}-\mathrm{e}^{-z}}{\mathrm{e}^{z}+\mathrm{e}^{-z}}$$

$$\tanh'(z) = 1-y^2$$

由此可见，激活函数的导数可以用原函数进行表示，这样只需要计算出原函数的值，就可

以直接计算其导数，从而降低了计算量。

2. 长短时记忆网络学习参数说明。

长短时记忆网络学习参数表如表 9-1 所示。

表 9-1　长短时记忆网络学习参数表

学习参数	权值矩阵	偏置项
遗忘门	$\boldsymbol{W}_f$	b_f
输入门	$\boldsymbol{W}_i$	b_i
输出门	$\boldsymbol{W}_o$	b_o
单元状态	$\mathbf{W}_c$	b_c

由于权值矩阵的两部分在反向传播中使用了不同的公式，故对以上四个权值矩阵分别进行拆分，拆分结果如下：

$$\boldsymbol{W}_f = [\boldsymbol{W}_{fs} \quad \boldsymbol{W}_{fx}]$$

$$\boldsymbol{W}_i = [\boldsymbol{W}_{is} \quad \boldsymbol{W}_{ix}]$$

$$\boldsymbol{W}_o = [\boldsymbol{W}_{os} \quad \boldsymbol{W}_{ox}]$$

$$\boldsymbol{W}_c = [\boldsymbol{W}_{cs} \quad \boldsymbol{W}_{cx}]$$

上面四组的拆分说明在后文会详细介绍。

3. 逐元素乘符号：∘

作用于两向量时为两个向量每个对应位置的元素相乘，维度保持不变。

向量与矩阵逐元素相乘时为列向量中的每个元素按列乘以矩阵中对应列的元素，并作用于矩阵的各列，此时向量的维度与矩阵的行数相同。乘积的结果为矩阵，此矩阵的维度与相乘之前的矩阵维度一致。

两矩阵逐元素相乘时为两个矩阵对应位置的元素相乘，此时两个矩阵维度相同。需要补充的是，逐元素相乘计算有时候可以简化矩阵和向量的计算。例如，对角矩阵右乘一个矩阵等于由这个对角矩阵中对角线上的各个元素组成的向量逐元素乘以那个矩阵，表示为：$\mathrm{diag}[a]\boldsymbol{X} = a \circ X$。一个行向量右乘一个对角矩阵等于这个向量逐元素乘那个矩阵对角线元素组成的向量，表示为：$a^{\mathrm{T}} \cdot \mathrm{diag}[b] = a \circ b$。

接下来具体每个步骤的详细计算处理方式如下。

1. 前向计算过程

前向计算即通过输入计算得到输出，汇总各类门的计算公式如下。

遗忘门计算方式：$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_f)$

输入门计算方式：$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_i)$、$\widetilde{\boldsymbol{c}_t} = \tanh(\boldsymbol{W}_c[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_c)$

单元状态更新计算方式：$\boldsymbol{c}_t = \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \widetilde{\boldsymbol{c}_t}$

输出门计算方式：$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_o)$、$\boldsymbol{s}_t = \boldsymbol{o}_t \circ \tanh(\boldsymbol{c}_t)$

x、**f**、**i**、**c**、**o**、**b**、$\boldsymbol{s}$均为向量表示，$\boldsymbol{W}$为权值矩阵。有两点需要注意：首先，$\boldsymbol{W}_f$、$\boldsymbol{W}_i$与$\boldsymbol{W}_c$、$\boldsymbol{W}_o$分别为遗忘门、输入门和输出门权值矩阵，b_f、b_i与b_c、b_o分别为其对一个的偏置项，矩阵之间、偏置项之间是不共享的；其次，由于输入向量的连接操作，以上输入向量与输出向量的维度是不相同的，具体分析如下。

定义当前输入层向量的维度为d_x，单元状态向量的维度为d_c，隐藏层向量的维度为d_s，通常d_c等于d_s，将输入门按照如下公式展开：

$$\begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \vdots \\ \boldsymbol{f}_{dc} \end{bmatrix} = \sigma\left(\begin{bmatrix} \boldsymbol{w}_{11} & \cdots & \boldsymbol{w}_{1(ds)} & \boldsymbol{w}_{1(ds+1)} & \cdots & \boldsymbol{w}_{1(ds+dx)} \\ \boldsymbol{w}_{21} & \cdots & \boldsymbol{w}_{2(ds)} & \boldsymbol{w}_{2(ds+1)} & \cdots & \boldsymbol{w}_{2(ds+dx)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{w}_{(dc)1} & \cdots & \boldsymbol{w}_{(dc)(ds)} & \boldsymbol{w}_{(dc)(ds+1)} & \cdots & \boldsymbol{w}_{(dc)(ds+dx)} \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{s}_{ds} \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{dx} \end{bmatrix} + \begin{bmatrix} b_{f(1)} \\ b_{f(2)} \\ \vdots \\ b_{f(dc)} \end{bmatrix} \right)$$

上式中的f_i表示第i个元素的Sigmoid函数输出，$\boldsymbol{W}_f$矩阵可以看作是由两个矩阵拼接而成，即与上一时刻输出向量$\boldsymbol{s}_{t-1}$进行计算的权值矩阵$\boldsymbol{W}_{fs}$和与当前输入向量$\boldsymbol{x}_t$进行运算的权值矩阵$\mathbf{W}_{fx}$，不难得出W_f的维度为$d_c \times (d_s + d_x)$。因此公式右侧部分可以按照如下方式进行简化：$\boldsymbol{W}_f[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_f = [\boldsymbol{W}_{fs} \quad \boldsymbol{W}_{fx}] \begin{bmatrix} \boldsymbol{s}_{t-1} \\ x_t \end{bmatrix} + b_f = \boldsymbol{W}_{fs}\boldsymbol{s}_{t-1} + \boldsymbol{W}_{fx}\boldsymbol{x}_t + b_f$ 。其他公式的连接操作运算也是按照以上规则处理即可。

理解以上各式后，即可完成向前计算。计算的主要目的有两个，一个是通过当前时刻输入值和上一时刻输出值对单元状态进行更新，另外一个是有选择地产生当前神经元的对外输出和下一时刻的状态输出，不同于传统循环神经网络的是当前输出了两组信息。

2. 误差项计算

在计算误差项之前，定义长短时记忆网络在t时刻的输出为$\boldsymbol{s}_t$，对应的误差项δ_t为：

$$\delta_t \overset{\text{def}}{\equiv} \frac{\partial E}{\partial \boldsymbol{s}_t}$$

可见当前时刻的误差项为损失函数对输出值的导数，这是由于长短时记忆网络有四个加权输入，分别为$\boldsymbol{f}_t$、$\boldsymbol{i}_t$、$\tilde{\boldsymbol{c}}_t$、$\boldsymbol{o}_t$，对其加权输入的定义如下：

$$\text{net}_{f,t} = \boldsymbol{W}_f[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_f) = \boldsymbol{W}_{fs}\boldsymbol{s}_{t-1} + \boldsymbol{W}_{fx}\boldsymbol{x}_t + b_f$$

$$\text{net}_{i,t} = \boldsymbol{W}_i[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_i) = \boldsymbol{W}_{is}\boldsymbol{s}_{t-1} + \boldsymbol{W}_{ix}\boldsymbol{x}_t + b_i$$

$$\text{net}_{\tilde{c},t} = \boldsymbol{W}_{\tilde{c}}[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_{\tilde{c}}) = \boldsymbol{W}_{\tilde{c}s}\boldsymbol{s}_{t-1} + \boldsymbol{W}_{\tilde{c}x}\boldsymbol{x}_t + b_{\tilde{c}}$$

$$\text{net}_{o,t} = \boldsymbol{W}_o[\boldsymbol{s}_{t-1}, \boldsymbol{x}_t] + b_o) = \boldsymbol{W}_{os}\boldsymbol{s}_{t-1} + \boldsymbol{W}_{ox}\boldsymbol{x}_t + b_o$$

进而得出对应的误差定义：

$$\delta_{f,t} \overset{\text{def}}{=} \frac{\partial E}{\partial \text{net}_{f,t}}$$

$$\delta_{i,t} \overset{\text{def}}{=} \frac{\partial E}{\partial \text{net}_{i,t}}$$

$$\delta_{\tilde{c},t} \overset{\text{def}}{=} \frac{\partial E}{\partial \text{net}_{\tilde{c},t}}$$

$$\delta_{o,t} \overset{\text{def}}{=} \frac{\partial E}{\partial \text{net}_{o,t}}$$

误差项按照时间的反向传递，即计算$t-1$时刻的误差项δ_{t-1}。δ_{t-1}的表达式为$\delta_{t-1} = \delta_{f,t}^{\mathrm{T}}\boldsymbol{W}_{fs} + \delta_{i,t}^{\mathrm{T}}\boldsymbol{W}_{is} + \delta_{\tilde{c},t}^{\mathrm{T}}\boldsymbol{W}_{\tilde{c}s} + \delta_{o,t}^{\mathrm{T}}\boldsymbol{W}_{os}$，根据$\delta_{f,t}^{\mathrm{T}}$、$\delta_{i,t}^{\mathrm{T}}$、$\delta_{\tilde{c},t}^{\mathrm{T}}$、$\delta_{o,t}^{\mathrm{T}}$的定义可知：

$$\delta_{f,t}^{\mathrm{T}} = \delta_t^{\mathrm{T}} \circ o_t \circ (1 - \tanh(c_t)^2) \circ c_{t-1} \circ f_t \circ (1 - f_t)$$

$$\delta_{i,t}^{\mathrm{T}} = \delta_t^{\mathrm{T}} \circ o_t \circ (1 - \tanh(c_t)^2) \circ \tilde{c}_t \circ i_t \circ (1 - i_t)$$

$$\delta_{\tilde{c},t}^{\mathrm{T}} = \delta_t^{\mathrm{T}} \circ o_t \circ (1 - \tanh(c_t)^2) \circ i_t \circ (1 - \tilde{c}_t^{\,2})$$

$$\delta_{o,t}^{\mathrm{T}} = \delta_t^{\mathrm{T}} \circ \tanh(c_t) \circ o_t \circ (1 - o_t)$$

以上就是将误差从当前时刻传播到前一时刻的公式。这样就可以得出任意时刻k的公式：

$$\delta_k = \prod_{j=k}^{t-1} (\delta_{f,j}^{\mathrm{T}}\boldsymbol{W}_{fs} + \delta_{i,j}^{\mathrm{T}}\boldsymbol{W}_{is} + \delta_{\tilde{c},j}^{\mathrm{T}}\boldsymbol{W}_{\tilde{c}s} + \delta_{o,j}^{\mathrm{T}}\boldsymbol{W}_{os})$$

在上述公式基础之上，可以通过当前时刻的误差项反向计算之前任意时刻的误差项的值。误差项按照结构由当前层l传递到上一层$l-1$的计算方法为，激活函数对$l-1$层加权输入的导数，表示如下：

$$\delta_t^{l-1} \overset{\text{def}}{=} \frac{\partial E}{\text{net}_t^{l-1}} = (\delta_{f,t}^{\mathrm{T}} \boldsymbol{W}_{fx} + \delta_{i,t}^{\mathrm{T}} \boldsymbol{W}_{ix} + \delta_{\tilde{c},t}^{\mathrm{T}} \boldsymbol{W}_{\tilde{c}x} + \delta_{o,t}^{\mathrm{T}} \boldsymbol{W}_{ox}) \circ f'(\text{net}_t^{l-1})$$

以上就是误差传到上一层的公式，这样就可以计算出每一层的误差项的值。

3. 权值梯度的计算与更新

权值梯度的计算方式为损失函数对权值矩阵的导数，最终梯度为各个时刻梯度之和。有了两个方向上误差项的值，就可以计算出各个权值矩阵的梯度。

1. 首先给出按照时间反向传播的梯度矩阵$\boldsymbol{W}_{fs}$、$\boldsymbol{W}_{is}$、$\boldsymbol{W}_{cs}$、$\boldsymbol{W}_{os}$的更新公式：

$$\frac{\partial E}{\partial \boldsymbol{W}_{fs,t}} = \frac{\partial E}{\partial \text{net}_{f,t}} \frac{\partial \text{net}_{f,t}}{\partial \boldsymbol{W}_{fs,t}} = \delta_{f,t} s_{t-1}^{\mathrm{T}}$$

$$\frac{\partial E}{\partial \boldsymbol{W}_{is,t}} = \frac{\partial E}{\partial \text{net}_{i,t}} \frac{\partial \text{net}_{i,t}}{\partial \boldsymbol{W}_{is,t}} = \delta_{i,t} s_{t-1}^{\mathrm{T}}$$

$$\frac{\partial E}{\partial \boldsymbol{W}_{cs,t}} = \frac{\partial E}{\partial \text{net}_{\tilde{c},t}} \frac{\partial \text{net}_{\tilde{c},t}}{\partial \boldsymbol{W}_{cs,t}} = \delta_{\tilde{c},t} s_{t-1}^{\mathrm{T}}$$

$$\frac{\partial E}{\partial \boldsymbol{W}_{os,t}} = \frac{\partial E}{\partial \text{net}_{o,t}} \frac{\partial \text{net}_{o,t}}{\partial \boldsymbol{W}_{os,t}} = \delta_{o,t} s_{t-1}^{\mathrm{T}}$$

2. 整体求和即可得出按照时间反向传播的四个矩阵的梯度公式：

$$\frac{\partial E}{\partial \boldsymbol{W}_{fs}} = \sum_{j=1}^{t} \delta_{f,j} \boldsymbol{s}_{j-1}^{\mathrm{T}}$$

$$\frac{\partial E}{\partial \boldsymbol{W}_{is}} = \sum_{j=1}^{t} \delta_{i,j} \boldsymbol{s}_{j-1}^{\mathrm{T}}$$

$$\frac{\partial E}{\partial \boldsymbol{W}_{cs}} = \sum_{j=1}^{t} \delta_{\tilde{c},j} \boldsymbol{s}_{j-1}^{\mathrm{T}}$$

$$\frac{\partial E}{\partial \boldsymbol{W}_{os}} = \sum_{j=1}^{t} \delta_{o,j} \boldsymbol{s}_{j-1}^{\mathrm{T}}$$

3. 同理可以得到偏置项的梯度计算公式：

$$\frac{\partial E}{\partial b_f} = \sum_{j=1}^{t} \delta_{f,j}$$

$$\frac{\partial E}{\partial b_i}=\sum_{j=1}^{t}\delta_{i,j}$$

$$\frac{\partial E}{\partial b_c}=\sum_{j=1}^{t}\delta_{\tilde{c},j}$$

$$\frac{\partial E}{\partial b_o}=\sum_{j=1}^{t}\delta_{o,j}$$

④ 然后计算按结构反向传播到上一层权值矩阵$\boldsymbol{W}_{fx}$、$\boldsymbol{W}_{ix}$、$\boldsymbol{W}_{cx}$、$\boldsymbol{W}_{ox}$的更新公式：

$$\frac{\partial E}{\partial \boldsymbol{W}_{fx}}=\frac{\partial E}{\partial \mathrm{net}_{f,t}}\frac{\partial \mathrm{net}_{f,t}}{\partial \boldsymbol{W}_{fx}}=\delta_{f,t}\boldsymbol{x}_t^{\mathrm{T}}$$

$$\frac{\partial E}{\partial \boldsymbol{W}_{ix}}=\frac{\partial E}{\partial \mathrm{net}_{i,t}}\frac{\partial \mathrm{net}_{i,t}}{\partial \boldsymbol{W}_{ix}}=\delta_{i,t}\boldsymbol{x}_t^{\mathrm{T}}$$

$$\frac{\partial E}{\partial \boldsymbol{W}_{cx}}=\frac{\partial E}{\partial \mathrm{net}_{\tilde{c},t}}\frac{\partial \mathrm{net}_{\tilde{c},t}}{\partial \boldsymbol{W}_{cx}}=\delta_{\tilde{c},t}\boldsymbol{x}_t^{\mathrm{T}}$$

$$\frac{\partial E}{\partial \boldsymbol{W}_{ox}}=\frac{\partial E}{\partial \mathrm{net}_{o,t}}\frac{\partial net_{o,t}}{\partial \boldsymbol{W}_{ox}}=\delta_{o,t}\boldsymbol{x}_t^{\mathrm{T}}$$

这样就得到了两个方向上权值矩阵梯度的计算方法，然后采用以上公式更新各个权值矩阵即可完成训练的一轮更新操作。

9.4.5　相关变种简介

Cho 等人在 2014 年提出了 Gated Recurrent Unit（GRU），GRU 是当前比较流行的一种长短时记忆网络的变种形式。GRU 的做法是将单元状态与隐藏状态进行合并，然后将原来的三个门替换为更新门（Update Gate）和重置门（Reset Gate），其结构如图 9-15 所示。

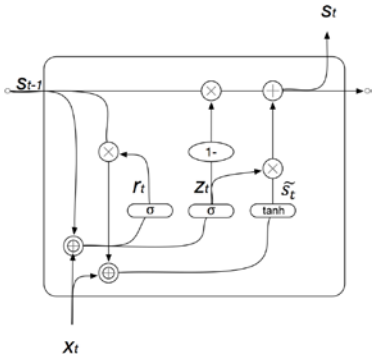


图 9-15　GRU 内部结构图

过图 9-15 中各门的各个状态表示公式如下：

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r \cdot [\boldsymbol{s}_{t-1}, \boldsymbol{x}_t])$$

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z \cdot [\boldsymbol{s}_{t-1}, \boldsymbol{x}_t])$$

$$\widetilde{\boldsymbol{s}_t} = \tanh(\boldsymbol{W} \cdot [\boldsymbol{r}_t * \boldsymbol{s}_{t-1}, \boldsymbol{x}_t])$$

$$\boldsymbol{s}_t = (1 - \boldsymbol{z}_t) * \boldsymbol{s}_{t-1} + \boldsymbol{z}_t * \widetilde{\boldsymbol{s}_t}$$

通过上述公式可以得出，重置门$\boldsymbol{r}_t$的主要作用是决定新的隐藏状态$\widetilde{\boldsymbol{s}_t}$中上一个时刻输出状态$\boldsymbol{s}_{t-1}$所占的比重。新的隐藏状态$\widetilde{\boldsymbol{s}_t}$由重置门过滤后的上一时刻输入$\boldsymbol{s}_{t-1}$和当前时刻输入$\boldsymbol{x}_t$通过 Tanh激活函数非线性变换后得到。更新门$\boldsymbol{z}_t$主要影响当前隐藏状态$\boldsymbol{s}_t$在新的隐藏状态$\widetilde{\boldsymbol{s}_t}$中所占的比重。同时控制了最终的输出$\boldsymbol{s}_t$中$\widetilde{\boldsymbol{s}_t}$和$\boldsymbol{s}_{t-1}$所占的比重。相比传统的长短时记忆网络，GRU 在结构上更加精简。除此之外，还有很多其他变种形式，包括但不限于如下：

（1）无输入门（No Input Gate，NIG）。

（2）无遗忘门（No Forget Gate，NFG）。

（3）无输出门（No Output Gate，NOG）。

（4）无输入激活函数（No Input Activation Function，NIAF），即没有输入门对应的tanh层。

（5）无输出激活函数（No Output Activation Function，NOAF），即没有输出门对应的$t$anh层。

（6）无“peephole connection”（No Peepholes，NP）。

（7）遗忘门与输出门结合（Coupled Input and Forget Gate，CIFG）。

9.5 常见循环神经网络结构

9.5.1 *N*比 *N*结构

N 比 *N* 结构表示循环神经网络 *N* 维的输入以及 *N* 维的输出，这种结构从传统的神经网络顺势衍生而来，是较为经典的循环神经网络结构，大致结构如图 9-16 所示。

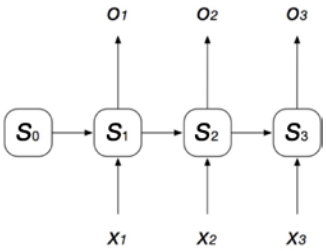


图 9-16　*N*比 *N*循环神经网络结构示意图

由于限制必须输入和输出的维度必须保持一致，因此其常常用于处理以下问题。

（1）视频理解中获取视频每一帧标签，输入为视频解码后的图像，通过此结构，获取每一帧的标签信息。这种场景一般用作视频理解的初期，对视频做初步的处理后，后续可以基于这些标签信息进行语义分析，构建更为复杂的需求场景。

（2）股票价格预测。基于历史的股票信息输入，预测下一时刻或者未来的股票走势信息。

9.5.2　*N*比 1 结构

N 比 1 结构表示循环神经网络 *N* 维的输入以及 1 维的输出。有时候需要对多种信息进行归纳汇总，这种场景下就需要一种网络结构，能够针对多样化的信息进行处理，并生成汇总信息，这种模型对应的则为循环神经网络的 *N* 比 1 结构，大致结构如图 9-17 所示。

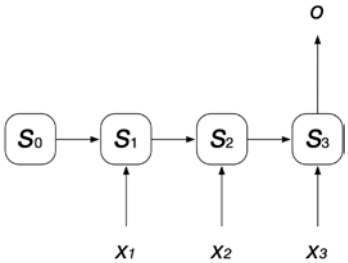


图 9-17　*N*比 1 循环神经网络结构示意图

N 比 1 结构的循环神经网络可以用于解决如下问题。

（1）视频理解中的获取视频每个场景的描述信息，或者获取整个影片的摘要信息。

（2）获取用户评价的情感信息，即根据用户的一句话的评论，来判断用户的喜好等情感信息。

9.5.3　1 比 *N*结构

1 比 *N* 结构表示循环神经网络 1 维的输入以及 *N* 维的输出。当输入是单维度而输出信息是多维度时，可以在序列开始的时候进行输入计算，后期的每次迭代不再使用外部输入，而是在

内部隐藏层之间进行信息的传递，最后将循环处理的结果输出，其大致结构如图 9-18 所示。

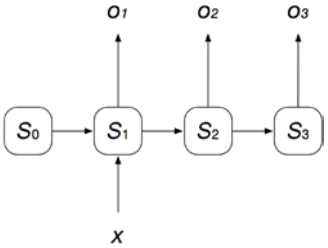


图 9-18　1 比 *N* 循环神经网络结构示意图（a）

除如图 9-18 所示外，还有一种是将同一信息不同时刻输入进循环神经网络，进行处理输出，如图 9-19 所示。

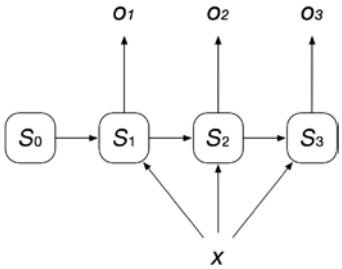


图 9-19　1 比 *N* 循环神经网络结构示意图（b）

这种 1 比 *N* 的结构，可以处理如下问题。

（1）看图写描述：根据输入的一张图片，生成对这张图片的描述信息。

（2）自动作曲：按照类别生成音乐。

9.5.4　*N* 比 *M* 结构

N 比 *M* 结构表示循环神经网络 *N* 维的输入以及 *M* 维的输出。这种结构又被称作 Encoder-Decoder 模型，也可以称之为 Seq2Seq 模型，这种模型的输入与输出可以不相等。该结构由两部分组成：编码部分与解码部分，整体结构如图 9-20 所示。

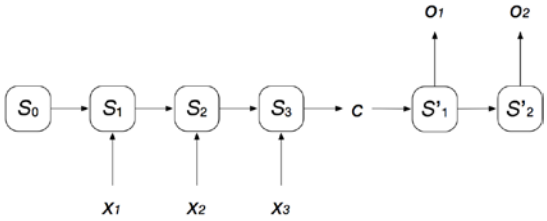


图 9-20　*N* 比 *M* 循环神经网络结构示意图（a）

$\boldsymbol{c}$的前半部分循环神经网络为编码部分，称之为 Endcoder，$\boldsymbol{c}$可以是$\boldsymbol{S}_3$的直接输出，或者是对s_3输出做一定的变换，也可以对编码部分所有的$\boldsymbol{S}_1$、$\boldsymbol{S}_2$、$\boldsymbol{S}_3$进行变换得到，这样$\boldsymbol{c}$中就包含了对$\boldsymbol{x}_1$、$\boldsymbol{x}_2$、$\boldsymbol{x}_3$的编码信息。$\boldsymbol{c}$的后半部分循环神经网络为解码部分，称之为 Decoder。$\boldsymbol{c}$作为之前的状态编码，作为初始值，输入到 Decoder 当中。Decoder 经过循环处理，最终将信息解码输出。从图 9-20 可见，输出的形式和个数与输入不一致，这类解码更多意义上是对输入信息做转义处理。除上面示意的解码结构外，还有如图 9-21 所示的结构。

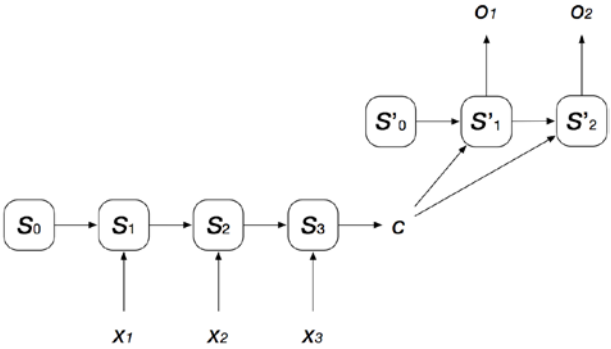


图 9-21　*N*比 *M*循环神经网络结构示意图（b）

N 比 *M* 的循环神经网络结构适用性更为普遍，因此现实中有很多基于这种结构的落地场景，它可以解决如下的问题。

（1）机器翻译：将不同语言作为输入，输出为非输入语言类别的语言，这也是 Encoder-Decoder 结构的经典应用场景。

（2）文本摘要：输入一篇文章，输出这篇文章的概要信息。

（3）语音识别：输入一段语音信息，输出这段语音信息的文字。

9.6　与自然语言处理结合

自然语言模型是针对自然语言处理过程中的语言表达模型，可以用于分词、信息抽取、词性标注、句法分析、情感分析、问答系统等。深度学习的众多算法均可以应用于自然语言处理领域，以卷积神经网络（CNN）和循环神经网络（RNN）作为对比，基于分层架构的前者在处理问题的识别和分类上，表现效果较好，而基于连续结构的后者更擅长使用其记忆功能对序列识别和建模。卷积神经网络与循环神经网络都可以应用在自然语言处理中。根据实际的任务，两者之间存在一定的互补优势，如表 9-2 所示。

表 9-2 卷积神经网络与循环神经网络在自然语言处理场景中的对比表

	卷积神经网络（CNN）	循环神经网络（RNN）
情感分析	–	–
语言建模	×	√
句子配对	√	×
命名实体识别	×	√
上下文问答	×	√

除上述的场景对比外，在不同句子长度的处理上，模型的效果与实际的句子长度相关度较大。在短句的场景下，由于卷积神经网络的卷积功能，对句子的结构纵览能力较强。但在处理长句时，卷积神经网络只能处理窗口内信息，相邻窗口信息只能借助后一层卷积层进行信息融合。具体效果与对卷积窗口的大小与移动步长的选择等参数相关性较大，因此调参优化就成为一个比较重要的工作。对循环神经网络来说，上述问题不会太明显，但是循环神经网络的训练性能次于卷积神经网络，因此需要占用更多的时间训练。

9.7 实例：文本自动生成

本章已经完整介绍了循环神经网络的理论知识。有了理论铺垫后，本节我们通过 TensorFlow 搭建一个循环神经网络作为演示实例，供大家学习参考。本实例主要实现对新闻稿学习并进行自动文本生成的功能。训练集选择的是《人工智能市场监控报告》（数据来自百度德尔塔俱乐部，共 12 期数据），训练集中内容大致如下所示。

```
1. 美国多家科技公司发布 AI 系统伦理道德原则
【摘要】包括 IBM、微软、谷歌、亚马逊、FB 和苹果等科技巨头的信息技术产业理事会，发布了应用 AI 系统所必须遵守的伦理道德原则。
https：//www.axios.com/tech-companies-pledge-to-use-artificial-intelligence-responsibly-2500397351.html?utm_source=sidebar
2. 研究数据：美国人工智能发展领跑全球
【摘要】数据显示，美国目前是人工智能发展的绝对冠军。美国的人工智能公司是中国的 1.82 倍。从世界 AI 公司的总数上看，美国占据 42%，而中国位居第二，占 23%。两国打败了英国、澳大利亚、日本、瑞典、新加坡等发达国家。
http：//technode.com/2017/10/22/china-vs-us-ai/
3.硅谷与高校争夺 AI 人才
【摘要】AI 被称为今年最大的投资趋势，也是最大的招聘趋势。AI 是直到最近才开始成为主流计算机科学的主题，这个领域的专家数量相对较少。由于科技公司提供更加丰厚的报酬，学术界的专家数量正在减少。在斯坦福大学，已经有 4 位著名的 AI 研究人员在过去几年里离职。在华盛顿大学的 AI 中心，该部门的 20 位教授中有 6 人正在休假或暂时休假。
http：//tech.qq.com/a/20171025/026179.htm
……
```

上述全部训练内容约为 9 万字。采取三层长短时记忆网络结构，使用 11000 次迭代后生成的模型，进行模拟内容的生成，如下内容为生成的关于人工智能相关的新闻稿。

```
1．谷歌推出新款 VR 头盔
【摘要】微软推出新款 AR 芯片，在一种机器人。
https：//www.technonet.com/store/2017/07/28/ipss-thinas-ai-and-interline-
deter-ifans-are-arsint-sons-stact/
2．中国联通成为人工智能研究院
【摘要】在贯彻公司将与北京交通委员会巡义管理局在中国市场的人工智能发展战略研究院，在中国建立人工智能研究院，并与 AR AI 技术、智能系统，人工智能领域的技术领域的竞争的云服务。
http：//www.sohu.com/a/190301311_663005?loc=4&tag_id=48044?hear=id=50
3．百度地图与北京机场试点无人驾驶汽车
【摘要】英特尔与安徽省无人驾驶汽车在自动驾驶汽车技术，这种无人驾驶汽车和汽车厂商结合的自动驾驶汽车的合作，并将在人工智能技术和应用。该公司还在其他公司将开发了一个名为“Yrapil Lidior Satlare Colle Motel Insirets 的自动驾驶车辆上路。
http：//finance.sina.com.cn/it/2017-07-14/doc-ifymfqmq8015421.shtml
```

从上述生成的内容可以看出，长短时记忆网络基本上可以学习到整个文章的结构模式，如段落结构的“标题+摘要+URL”模式，特别是对 URL 模式的学习已经很准确了。对于文本模式的学习基本可以保证句子的些许连贯性，如“百度地图与北京机场试点无人驾驶汽车”这样的通顺语句。

同时还有很多问题，如上下文关联度不高、内容之间并无直接关系、语义的理解还不够。当然这与训练迭代的次数有一定关系，相信进行更多次训练后，可以达到更好的效果。

9.8　本章小结

本章从传统的神经网络与循环神经网络处理问题的不同场景入手，引入了循环神经网络的概念。首先从整体上介绍循环神经网络的分类及结构形态，然后给出这种带有循环层网络的通用训练方法 BPTT。通过训练公式分析，引出了梯度爆炸和梯度消失问题。为了解决此问题，本章给出了当前市场上最为成功的循环网络结构：长短时记忆网络，并针对长短时记忆网络做了深入详细的分析，并简单给出其变种供读者参考，方便进一步深入学习。

本章在完成了理论到实际应用网络结构的分析后，又给出了各种循环神经网络的结构模型，供读者在实际使用中参考。

第 10 章

深度信念网络

深层信念网络（Deep Belief Network，DBN）是由 Geoffrey Hinton 在 2006 年提出的一种生成式模型， 通过训练神经元之间的权值，可以让整个神经网络根据最大概率生成训练数据。可以使用深度信念网络进行特征识别、数据分类以及对数据进行统计建模，表征事物的抽象特征或统计分布。

10.1 概要

10.1.1 背景

常见的前馈型神经网络一般包括输入层、隐藏层和输出层。从理论的角度，倘若隐藏层的深度越深，则代表该神经网络的表达能力越强，但是在实际使用过程中，尤其是当隐藏层的深度设计较深时，采用随机的方式为权值进行初始化，使用梯度下降的方式进行参数优化则会面临诸多问题。例如，权值的初始化倘若过大，则模型训练过程中权值会陷入局部最小值，而不是期望的全局最小值；相反，如果权值的初始化过小，则当使用反向传播进行参数调整时，梯度值则会很小，从而影响权值的改变也很小，最终可能会导致权值无法获得最优值。

理想的状态是权值在初始化的过程中，已经比较接近最优值，然后再使用梯度下降则可以得到一个比较好的模型结果，不仅效果好，而且收敛速度更快。Geoffrey Hinton 等在 2006 年提出了新的方式以求得获得最优解的权值初始化方法。

深度信念网络目前已经在语音识别、图像识别以及自然语言处理等领域取得了较好的效果，并得到了广泛的应用。

10.1.2 基本结构

深度信念网络的结构是由层叠的多个受限波尔兹曼机构成的一种产生式模型，并使用梯度下降法和反向传播算法对模型进行调优。简单的深度信念网络的拓扑结构如图 10-1 所示，通

过训练的深度信念网络，可以学习到训练样本数据的概率分布。

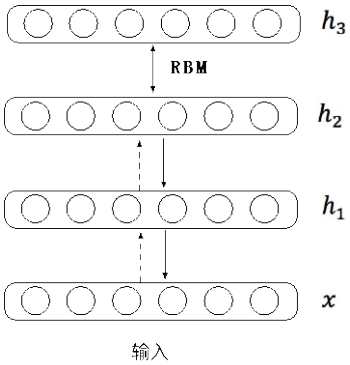


图 10-1　简单的深度信念网络结构示例

图 10-1 中的x表示可视层，h_1、h_2、h_3表示隐藏层。深度信念网络中有众多神经元，其中比较重要的两个为显性神经元（显元）和隐性神经元（隐元）。显元用于接受输入，而隐元用于提取特征。因此，隐元也被称作特征检测器或功能探测器。深度信念网络的前两层之间连接是不可见的，形成联想记忆。下层连接在上部和下部之间。底层代表数据向量，每个神经元代表数据向量的一个维度。

深度信念网络包括无监督的深度信念网络和有监督的深度信念网络两种，图 10-1 表示的是无监督的深度信念网络，有监督的深度信念网络如图 10-2 所示。

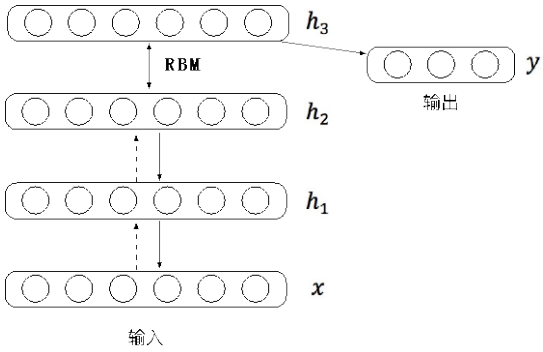


图 10-2　有监督的深度信念网络结构示例

深度信念网络可以基于无监督学习的方式对未标记的数据进行聚类，也可以在受限玻尔兹曼机层的堆叠结构最后加上一个 Softmax 层来构成分类器，成为分类模型。

10.2 受限玻尔兹曼机

10.2.1 概述

玻尔兹曼机（Boltzmann Machine，BM）是 Geoff Hinton 和 Sejnowski 于 1986 年基于统计力学提出的神经网络模型，玻尔兹曼机中的神经元是随机神经元，神经元的输出只有抑制状态和激活状态两种，一般用二进制 0 和 1 表示，其结构如图 10-3 所示。

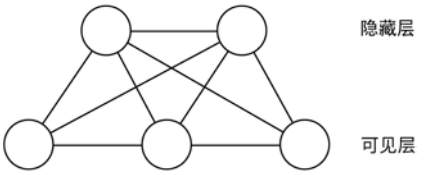


图 10-3 玻尔兹曼机的结构示例

标准的玻尔兹曼机是一个完全连接的图结构，具有较强的无监督学习能力，能够从训练样本中获得数据之间的规则，但是正是这种完全的图结构，导致计算复杂度很高，训练周期过长且很难获得概率分布，难以解决实际问题。因此，为优化玻尔兹曼机的结构以解决实际问题，在玻尔兹曼机的基础上引出了受限玻尔兹曼机（Restricted Boltzmann Machine，RBM）。受限玻尔兹曼机最初由 Paul Smolensky 于 1986 年提出，2006 年 Hinton 等人发明了快速学习算法后，才使得受限玻尔兹曼机逐渐广为人知，它是一种可以通过输入训练数据集概率分布的随机生成神经网络，其结构如图 10-4 所示。

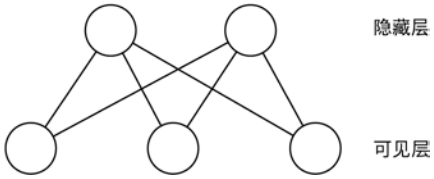


图 10-4 受限玻尔兹曼机的结构示例

受限玻尔兹曼机在结构上移除了同一层之间的连接（这就是受限玻尔兹曼机中“受限”的含义），层内没有任何节点相互关联。受限玻尔兹曼机是浅层的、构成深度信念网络构建的两层神经网络，在结构上第一层为可见层，第二层为隐藏层。

受限玻尔兹曼机是一种适用于降维、分类、回归、协同过滤、特征学习以及主题建模的模型，其整体结构相对简单，但具有很强的应用性。

每个可见层的节点都从训练的数据集的样本中获取低层次的特征。例如，从灰度图像的数据集中每个可见层节点将接收每一个图像的每一个像素值。以 MNIST 图像为例，MNIST 图像

长宽均为 28，因此每张图像都有 784 像素，所以处理它们的神经网络在可见层上具有 784 个输入节点，保证每一个像素对应一个可见层节点。

对于某个像素值 x，均包含可见层和隐藏层的两层网络。在隐藏层的节点处，将 x 乘以权值并加上偏置项得到的结果传递给激活函数，该激活函数产生节点的输出或通过它的信号的强度。

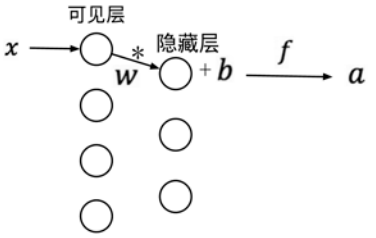


图 10-5　受限玻尔兹曼机的向前传递示例

在图 10-5 所示的结构中，用公式表示则为$a = f(x * w + b)$。从公式结构上看，其与感知器的前向传播方式类似，但是应尽量与感知器分隔开，因为两者的理论基础不同。感知器采用的是基于决策函数的变量转换，而受限玻尔兹曼机采用的是基于概率分布的变量转换。由于所有可见层的节点输入都被传递到所有隐藏层的节点，所以可以将受限玻尔兹曼机定义为对称的二分图。对称结构意味着每个可见层节点与每个隐藏层的节点相连。二分则意味着它具有多个层。每个隐藏层的节点，均是每个输入x乘以其相应的权值w并加上偏置项。完整的结构如图 10-6 所示。

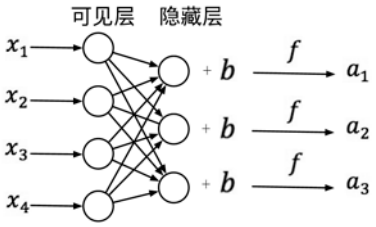


图 10-6　受限玻尔兹曼机的节点计算示例

图 10-6 输出的a_1、a_2、a_3也可以再次传递，如图 10-7 所示。

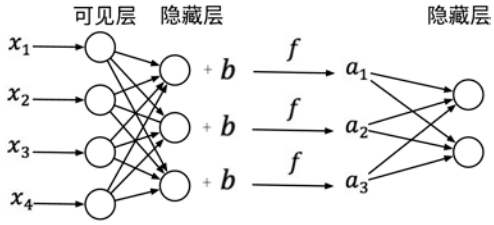


图 10-7　受限玻尔兹曼机的输出再次传递示例

传统的受限玻尔兹曼机都是基于二值离散变现的模型训练，但是在实际工作中，更多的是连续变量，因此衍生出了连续受限玻尔兹曼机（Continuous Restricted Boltzmann Machine，CRBM），它是受限玻尔兹曼机的一种形式。有效的连续受限玻尔兹曼机对可见层采用高斯变换，并在隐藏层上采用修正线性单元变换。对于处理二进制数据的受限玻尔兹曼机，可见层和隐藏层均采用二进制变换。高斯变换在受限玻尔兹曼机的隐藏层上效果相对较差，所使用的修正线性单元变换能够表示的特征比二进制变换更多。除连续受限玻尔兹曼机外，受限玻尔兹曼机还与其他结构结合，形成新的模型。例如，与卷积结合的卷积受限玻尔兹曼机。受限玻尔兹曼机实质是玻尔兹曼机和马尔科夫随机场的一类特例。

受限玻尔兹曼机可以根据具体的任务形式，选择使用有监督的学习方式或者无监督的学习方式。受限玻尔兹曼机在神经网络中有两个重要的作用：一方面是对数据进行编码，编码可以理解为特征抽象的过程，例如将降维得到的结果交给有监督的学习方法对数据进行分类。另一方面则是通过训练受限玻尔兹曼机获得神经网络的初始化权值矩阵和偏置项，然后供神经网络训练。

10.2.2 逻辑结构

受限玻尔兹曼机的数据重构主要是针对输入的数据，重构的核心在于有效地提取输入数据的特征，并构建新的数据结构进行后续分析。一般的神经网络存在两个基本功能：特征变换或回归预测，每个神经网络都具备二者之一。受限玻尔兹曼机主要是进行特征变换，这种变换即是对输入数据的重构过程。对于输入的特征X，进行转换之后得到更明晰的特征Y，受限玻尔兹曼机的重要任务则是找到特征X与特征Y的映射关系，这种映射关系用联合概率分布$P(X,Y)$表示。如果已经知道$P(X,Y)$的具体形式，则可以通过极大似然函数求解概率分布的参数。若不知道$P(X,Y)$具体形式，而仅知道变量之间的依赖关系，则可以通过能量模型（Energy Based Model）构建概率分布。

对于一个两层的玻尔兹曼机，设定i为可见层神经元的数量，可见层的神经元用v_i表示，j为隐藏层神经元的数量，隐藏层的神经元用h_j表示，$\boldsymbol{W}$为$i*j$的矩阵，表示可见层与隐藏层神经元之间的权值，同时设定对于可见层每一个神经元的v_i偏置为a_i，而对于隐藏层中每一个神经元h_j的偏置为b_j，因此可以通过如下公式定义受限玻尔兹曼机的能量：

$$E(v,h) = -\sum_i a_i\, v_i - \sum_j b_j\, h_j - \sum_i \sum_j h_j \boldsymbol{W}_{ij} v_i$$

对于一般的玻尔兹曼机，能量函数为隐藏层和可见层之间的联合概率分布的重要参数，公式：

$$P(v,h) = \frac{1}{Z} \mathrm{e}^{-E(v,h)}$$

上述公式中，其中Z为配分函数，表示的是节点在所有可能的取值下$e^{-E(v,h)}$的值之和，通俗理解即为概率分布和为 1 的归一化常数。同理，可见层取值的边缘分布可以通过对所有隐藏层配置求和得到，公式如下：

$$P(v)=\frac{1}{Z}\sum_h -E(v,h)$$

受限玻尔兹曼机相对于玻尔兹曼机的结构不同之处在于，同层之间的神经元并没有产生连接，因此，隐藏层的神经元是否激活在给定可见层节点值的情况下是条件独立事件。同理，对于可见层神经元的激活状态在给定隐藏层值的情况下也是条件独立事件。因此，对于包含m个可见层节点以及n个隐藏层的节点，可见层v对于隐藏层h的条件概率公式为：

$$P(v|h)=\prod_{i=1}^{m}P(v_i|h)$$

相反，隐藏层h对于可见层v的条件概率公式为：

$$P(h|v)=\prod_{j=1}^{n}P(h_i|v)$$

而对于单个节点的激活概率公式如下，其中σ表示逻辑函数：

$$P\left(h_j=1\middle|v\right)=\sigma(b_j+\sum_{i=1}^{m}w_{ij}v_i)$$

$$P(v_i=1|h)=\sigma(a_i+\sum_{j=1}^{n}w_{ij}h_j)$$

受限玻尔兹曼机在第一个隐藏层有效地学习输入数据的结构之后，会逐步向网络中的下一层隐藏层传递，当传递到下一层隐藏层时，上一个隐藏层的作用则和第一个可见层相似。第一个隐藏层的激活函数输出值为第二个隐藏层的输入值，同理，第二个隐藏层也可以按照同样的方法依次向后传递。数据的特征在不同的层次结构之间不断被分组和更抽象的表达，将特征分组，然后将分组之后的特征再进行分组，逐渐形成特征的层次分析，这点和卷积神经网络的卷积和池化过程有一定的相似之处。

在受限玻尔兹曼机的多层结构中，每增加一个隐藏层，其权值都会进行迭代调节，直至该层能够有效地模拟出前一层的输入。这是一种无监督的逐层贪婪预定型方法，对未被标记的数据进行无监督的聚类。数据在未被标记的情况下，通过权值的方式模拟出数据的特征，并将数据划分为不同的类别，可以使得受限玻尔兹曼机帮助有监督学习的深度信念网络进行图像分类等任务。

受限玻尔兹曼机的应用非常广泛，其中一个比较好的功能是可以帮助权值进行合理的初始化，因为通过权值模拟了的数据特征可以有效地帮助神经网络模型进行训练。深度的受限玻尔兹曼机是一个有向无环图。

10.2.3 对比分歧算法

受限玻尔兹曼机的训练目的是最优化权值矩阵$\boldsymbol{W}$，若要获得最优化的权值矩阵$\boldsymbol{W}$，则需要针对某一训练集$\boldsymbol{V}$，获得最大化概率的乘积。$\boldsymbol{V}$表示一个矩阵，其中$\boldsymbol{v}$是$\boldsymbol{V}$的一个行向量，用公式表达则为： $\arg\max_{\boldsymbol{W}} \prod_{v \in \boldsymbol{V}} P(v)$。

受限玻尔兹曼机的训练算法包括 Gibbs 采样（Gibbs Sampling）、变分近似方法（Variational Approach）、对比分歧（Contrastive Divergence）以及模拟退火（Simulate Annealing）等。

在对比分歧算法出现之前，一般采用 Gibbs 采样，虽然 Gibbs 采样可以得到对数似然函数关于未知参数梯度的近似，但是这样会使得采用较大的采样部署，其后果是会导致受限玻尔兹曼机的训练效率依然低下，尤其在观测数据的特征维度相对较高时，因此，Hinton 提出了受限玻尔兹曼机中的一个快速学习算法，即对比分歧算法，使得训练效率得到有效提升。

对比分歧算法在梯度下降的过程中使用 Gibbs 采样完成对网络中权值的更新，与前馈型神经网络的反向传播算法进行权值调节类似。对于某一个样本的单步对比分歧的步骤大致如下。

1. 在训练集样本中取某一样本$\boldsymbol{v}$，计算隐藏层节点的概率，在这一概率分布中获取一个隐藏层节点激活向量的样本$\boldsymbol{h}$。
2. 计算样本$\boldsymbol{v}$与样本$\boldsymbol{h}$的外积$\boldsymbol{v}\boldsymbol{h}^{\mathrm{T}}$，将此结果称为正梯度。
3. 从h获取一个重构的可见层节点的激活向量样本$\boldsymbol{v}'$，从$\boldsymbol{v}'$再次获得一个隐藏层节点的激活向量$\boldsymbol{h}'$。
4. 计算$\boldsymbol{v}'$与$\boldsymbol{h}'$的外积$\boldsymbol{v}'\boldsymbol{h}'^{\mathrm{T}}$，将此结果称为负梯度。
5. 设定η为学习速率，使用正梯度和负梯度的差值并结合学习速率，调整更新权值$\boldsymbol{w}_{ij}$，用公式表示则为：$\Delta w_{ij} = \eta(\boldsymbol{v}\boldsymbol{h}^{\mathrm{T}} - \boldsymbol{v}'\boldsymbol{h}'^{\mathrm{T}})$。

10.3 训练过程

10.3.1 工作流程

在深度信念网络的训练过程中，采用了逐层无监督的方式逐层训练参数，如图 10-8 所示。

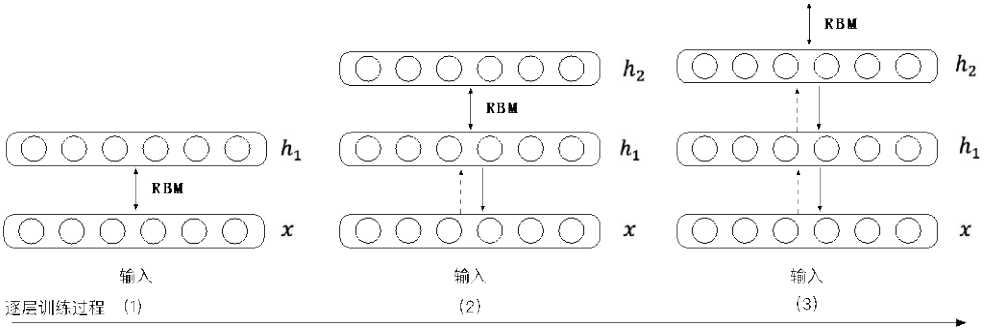


图 10-8　深度信念网络逐层训练的过程示例

如图 10-8 所示，首先把输入向量$\boldsymbol{X}$和第一个隐藏层h_1作为受限玻尔兹曼机，训练出当前玻尔兹曼机中的参数，包括向量$\boldsymbol{X}$与隐藏层h_1之间的权值，以及各个节点的偏置等。当获得当前受限波尔兹曼机参数之后，参数不再改变；然后将当前的隐藏层h_1作为可见层，将隐藏层h_2作为新的受限玻尔兹曼机，训练当前的受限玻尔兹曼机，并获得参数；同理，将当前的隐藏层h_2作为可见层，将隐藏层h_3作为新的受限玻尔兹曼机，再次训练，依次逐层计算。在深度置信网络中，受限玻尔兹曼机的对比分歧算法是进行层次训练的基础。

上述逐层进行训练的过程，在深度学习中被称作预训练过程，深度置信网络单独且无监督的训练每一层的受限玻尔兹曼机，通过层与层之间的映射，将特征向量映射到不同的特征空间，以使得特征尽可能多的保留。

有监督的深度信念网络一般是由若干层的受限玻尔兹曼机和一层全连接的神经网络组成的。最后一层受限玻尔兹曼机的隐藏层与全连接的网络连接，将最后一层受限玻尔兹曼机输出的特征向量作为全连接的输入向量，在全连接的网络中进行分类训练。每一层的受限玻尔兹曼机都通过预训练确保自身的网络层中对该层特征向量的映射已经达到最优，并不保证整个深度信念网络的特征表达最优，所以反向传播算法会将错误的信息由上而下传递至每一层的受限玻尔兹曼机。微调是深度学习训练过程的第二个阶段，第一阶段是预训练。受限玻尔兹曼机的预训练过程可以看作是前馈神经网络中的参数初始化过程，这样的预训练方式克服了由于随机参数导致的训练周期过长及模型陷入局部最优的问题。

10.3.2　调优过程

调优过程（Fine-Tuning）是一种训练算法的优化过程，生成模型一般采用 Contrastive Wake-Sleep 算法进行调优，大致可以分为三个过程。

1. 除顶层的受限玻尔兹曼机外，其他层的受限玻尔兹曼机的权值被划分为向上的认知权值以及向下的生成权值。

② 调优的 Wake 阶段。Wake 阶段是认知的过程，通过外部的数据特征和向上的认知权值产生每一层的抽象表示，并使用梯度下降的方式修改层与层之间的向下生成权值，实现的是尽可能与目标期望一致。

③ 调优的 Sleep 阶段。Sleep 阶段是生成的过程，通过顶层的表示以及想象权值，生成底层的状态，并修改层与层之间的向上认知权值。

10.4 本章小结

深度信念网络是较早的深度概率生成模型的典型代表，它由多层神经元构成，具有较好的特征学习能力和较快的模型训练能力。本章系统介绍了深度信念网络的产生背景和基本结构，并深入阐述了受限玻尔兹曼机的工作原理，详细介绍了受限玻尔兹曼机的对比分歧算法。受限玻尔兹曼机是深度信念网络组成的基础，可辅助进行参数预训练。

本章还介绍了深度信念网络的层次训练结构以及调优过程，通过预训练和微调使得模型在训练周期和训练效果上都达到一个较为理想的状态。

第 11 章

生成对抗网络

早在 19 世纪中叶，达尔文就在生物进化论中提出“物竞天择、适者生存”的对抗理念。人类有着强大的类比学习能力，正如我们参照飞翔的鸟类发明了飞机、参考鱼的外形发明了潜艇。如果将对抗引入机器学习，势必也会引起积极的连锁反应。

11.1 概述

生成对抗网络，顾名思义是一种通过对抗、竞争的方式生成数据的网络结构。不同于早期的机器学习模型，生成对抗网络首次将对抗的思想引入机器学习领域。Yann LeCun 认为对抗训练是最酷的事情：“Adversarial training is the coolest thing since sliced bread”。由此可见，生成对抗网络是一种充满发展前景的深度学习神经网络。

11.1.1 背景概要

普通神经网络可以分为“生成模型”和“判别模型”。判别模型试图在输入的特征之间建立一个近似关系，以达到对输入进行分类的目的。生成模型则用于模拟训练样本的概率分布，并试图生成与训练样本具有相同概率分布或相似特征的新样本。生成模型可用于图像清晰度提升、破损或遮挡图像的修复、样本数据生成等场景。

近几年深度学习取得的成就和影响力大多集中在判别模型，相比之下，生成模型的进展和突破则相对缓慢。这主要是因为传统的生成模型，如高斯混合模型、隐马尔科夫链模型等需要估计真实样本的概率分布，通过参数拟合出真实样本的概率分布后，再对分布随机采样，进而生成新样本。这其中最大的难点在于如何估计真实样本的概率分布。鉴于此，Ian Goodfellow 等人于 2014 年 6 月提出了生成对抗网络（Generative Adversarial Network，GAN）。

生成对抗网络主要解决的问题是：如何生成出符合真实样本概率分布的新样本。当输入的样本为图像时，生成对抗网络则生成与真实样本具有相似概率分布的图像；当输入的样本为文

本时，生成对抗网络生成与真实样本具有相似概率分布的文本；当输入的样本为语音时，GAN 则生成与真实样本具有相似概率分布的语音。

生成对抗网络自提出后，就获得了广泛的关注。截至 2017 年，短短的 3 年间，生成对抗网络相关的研究论文就如雨后春笋，不断涌现出新的思想和模型。

图 11-1 以时间为轴列举了几篇目前在生成对抗网络技术领域较具代表性的论文，这些论文几乎影响了生成对抗网络的发展趋势。

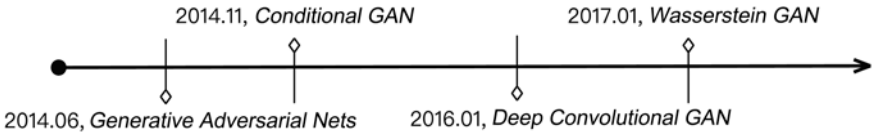


图 11-1 生成对抗网络的发展简要

（1）2014 年 6 月，Ian Goodfellow 发表了一篇名为 *Generative Adversarial Nets* 的论文，该文章第一次提出并描述了生成对抗网络。

（2）2014 年 11 月，Mehdi Mirza 将约束条件引入了生成对抗网络，发表了 *Conditional Generaive Adversarial Nets*，该文通过给生成对抗网络加上约束条件，使得生成的新样本符合预期。

（3）2016 年 1 月，Facebook AI 团队发表了 *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*，该文将深度卷积神经网络引入生成对抗网络，提出深度卷积生成对抗网络。引入卷积神经网络后，不仅能够加快生成对抗网络的训练过程，而且还使得训练过程更加稳定。

（4）2017 年 1 月，Martin Arjovsky 发表了 *Wasserstein GAN*，该文使用瓦瑟斯坦距离（Wasserstein 距离）取代 Jensen-Shannon 距离（JS 距离）来衡量生成样本和真实样本概率分布之间的距离，WGAN 能够生成具有多样性的新样本，基本解决了 GAN 存在的模型崩塌问题。

自 2016 年起，GAN 相关的研究论文和成果呈指数增长趋势，GAN 已然成为深度学习的下一个热门方向。

11.1.2 核心思想

生成对抗网络是一种包含无监督学习领域的人工智能算法，通过无监督的方式对真实样本的学习，模拟其数据分布的情况，以产生相似的样本数据为目的。其网络结构如图 11-2 所示。

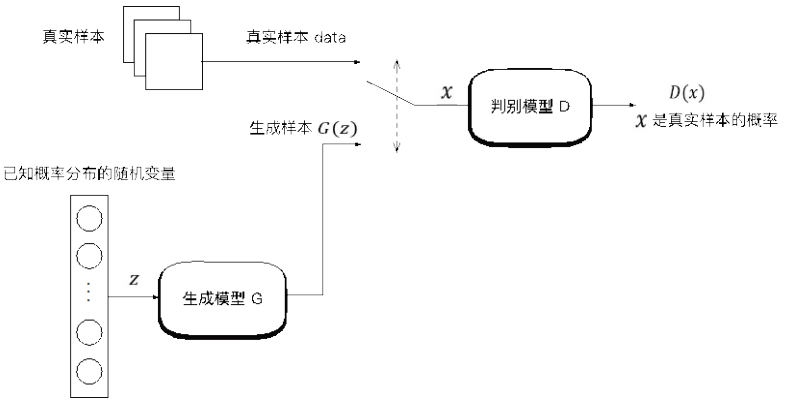


图 11-2　生成对抗网络基本结构

根据图 11-2，GAN 由生成模型 G 和判别模型 D 两个子模型组成，生成模型的目的是使生成的新样本与真实样本尽可能相似，而判别模型的目的则是尽量准确无误地区分真实样本和生成样本。在 GAN 提出之前，生成模型的主要思想是：模拟真实样本的概率分布。在获取真实样本概率分布的前提下，通过对其随机采样生成新的样本。鉴于获取真实样本的概率分布难度通常较大，GAN 提出一种新的思想：通过学习一组随机变量到真实样本的映射关系，进而获取一个由多层神经网络组成的模型。GAN 不直接估计真实样本的概率分布，而是通过模型学习的方式生成与真实样本具有相同概率分布的新样本。

11.1.3　基本工作流程

生成对抗网络的一般训练过程如下。

首先，固定生成模型，将真实样本作为初始数据输入判别模型，训练判别模型达到一定的判别准确度。

然后，固定判别模型，将一组随机变量输入生成模型，再将生成模型输出的生成样本输入判别模型进行判别。

在训练过程中，对生成模型和判别模型同时使用反向传播计算梯度，不断更新模型参数。在此过程中，生成模型和判别模型都会优化各自网络，判别模型能够越来越准确地区分真实样本和生成样本，而生成模型则会生成越来越接近真实样本的新样本。最终，双方达到一个动态平衡，即纳什均衡。达到动态平衡时，生成模型能够生成和真实样本几乎一模一样的新样本，即生成模型能够恢复真实样本的分布。此时，判别模型的判别结果为真实样本和生成样本各占 50%，再也无法区分出真实样本和生成样本。

因此，生成对抗网络可以描述为一个零和博弈问题，博弈双方正是生成模型和判别模型，

上述训练过程可用公式表述为：

$$\min_G \max_D V(D,G)$$

上述公式中，$V(D,G)=E_{x\sim p_{\text{data}}(x)}[\log D(x)]+E_{z\sim p_z(z)}[\log(1-D(G(z)))]$。$p_{\text{data}}$表示真实样本的概率分布；$z$表示服从已知分布的随机变量；$p_z$表示$z$的概率分布，常用的如高斯分布或均匀分布；$x\sim p_{\text{data}}(x)$表示$x$取自真实样本的分布；$z\sim p_z(z)$表示$z$取自我们模拟的分布；$G(z;\theta_g)$表示参数为$\theta_g$的生成模型，$G$是可微函数，以保证生成模型的误差能够反向传播；$D(x;\theta_d)$表示参数为$\theta_d$的判别模型，$D$是可微函数，以保证判别模型的误差能够反向传播。

其中，$G(z;\theta_g)$负责将随机变量映射到真实样本的分布，输出生成样本。$D(x;\theta_d)$可看作是一个二分类器，输出单变量用于表征x来自真实样本而非生成样本的概率。当固定生成模型时，对于判别模型的优化，可以这样理解：如果输入来自真实样本，则$D(x;\theta_d)$优化模型使输出尽可能接近 1；如果输入来自生成样本，则$D(x;\theta_d)$优化模型使输出尽可能接近 0。当固定判别模型时，$G(z;\theta_g)$优化模型输出尽可能和真实样本一模一样的新样本，并且使得生成样本经过$D(x;\theta_d)$的判别之后的输出尽可能接近 1。

由此可见，$V(D,G)$中，$\log D(x)$表示判别模型认为x来自真实样本的概率，$(1-D(G(z)))$表示判别模型认为$G(z)$来自生成样本的概率。因此，$\min_G \max_D V(D,G)$可理解如下，判别模型的训练目标可描述为：将输入的真实样本判别为真实样本、将输入的生成样本判别为生成样本的概率最大化，即$\max_D V(D,G)=E_{x\sim p_{\text{data}}(x)}[\log D(x)]+E_{z\sim p_z(z)}[\log(1-D(G(z)))]$；生成模型的训练目标可描述为，将输入判别模型的生成样本判别为生成样本的概率最小化，即$\min_G E_{z\sim p_z(z)}[\log(1-D(G(z)))]$。

生成对抗网络的训练过程并非将 $\min_G \max_D V(D,G)$看作优化目标，直接计算其对θ_g和θ_d的梯度，而是分为如下两个步骤。

1. 训练判别模型，更新θ_d。

将一组真实样本x以及一组生成样本$G(z;\theta_g)$输入判别模型，计算$\nabla_{\theta_d}\{E_{x\sim p_{\text{data}}(x)}[\log D(x)]+E_{z\sim p_z(z)}[\log(1-D(G(z)))]\}$，并采用梯度上升方法更新$\theta_d$，将上述过程循环$k$次。

2. 训练生成模型，更新θ_g。

将一组一维随机变量z输入生成模型，计算$\nabla_{\theta_g}\{E_{z\sim p_z(z)}[\log(1-D(G(z)))]\}$，并采用梯度下降方法更新$\theta_g$。

训练过程中，步骤1之所以会循环执行k次，主要是考虑让判别模型尽可能与生成模型匹配，即在判别模型达到一定判别准确度的基础上再训练生成模型。步骤2中计算梯度时省略

了$E_{x\sim p_{\text{data}}(x)}[\log D(x)]$，这是因为真实样本并非$\theta_g$的函数，因此真实样本对$\theta_g$的导数为零。

生成对抗网络及其训练过程可以通过赝品制作和文物鉴别这一活动做形象类比说明。其中，可以将制造赝品的作坊看作生成模型，将负责检测货品是赝品还是真迹的鉴定专家看作判别模型。赝品作坊的目标是想方设法制造出和文物真迹一模一样的赝品，使得专家无法区分出赝品和真迹；而鉴定专家的目标是想方设法检测出赝品和真迹，让赝品无所遁形。假设有两款号称官窑的青花瓷，一真一假。一开始，鉴定专家能够较好地区分出赝品，因为赝品的落款不正确。而作坊获知是由于落款的原因导致赝品被识破，他们就会改进造假手段。第二次，等新的赝品出炉后，鉴定专家或许又能发现其他瑕疵，进而给出赝品的判断结论。然后，作坊又一次完善造假手段。如此循环往复，在这个不断迭代的过程中，作坊丰富了造假手段、提高了赝品制造水平，几乎能够产出和青花瓷真迹相差无几的赝品；同时，鉴定专家的鉴定手段和水平也获得了提升，能够更加准确地区分出赝品和真迹。

11.2　朴素生成对抗网络

11.2.1　网络结构

简单生成对抗网络的生成模型和判别模型可通过全连接神经网络实现，本文称之为朴素生成对抗网络。对于朴素生成对抗网络，判别模型、生成模型和损失函数是其重要的组成部分。

1. 判别模型

基于简单的神经网络作为判别模型的结构大致如图 11-3 所示。

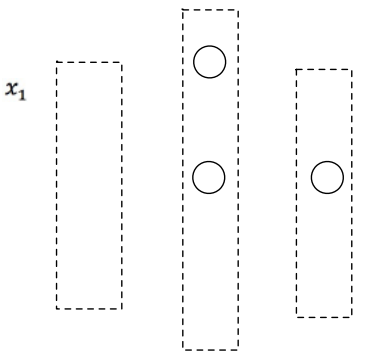


图 11-3　一个简单的神经网络作为判别模型示例

判别模型是由输入层、隐藏层、输出层组成的三层神经网络。该神经网络输入的是真实样

本或生成样本，输出的是当前样本为真实样本而非生成样本的概率。采用以下记号描述判别模型对应的神经网络：D_L表示神经网络的层数，D_n^l表示第l层神经元的数量，$D_\sigma_l(\cdot)$表示第l层神经元的激活函数，$D_W^l \in \mathbb{R}^{n^{l-1}\times n^l}$表示第$l-1$层到第$l$层神经元的权值矩阵，$D_b^l \in \mathbb{R}^{n^l}$表示第$l$层神经元的偏置量，$\in \mathbb{R}^{n^l}$表示第$l$层神经元的输入，$D_a^l \in \mathbb{R}^{n^l}$表示第$l$层神经元的输出。

2. 生成模型

基于简单的神经网络作为生成模型的结构大致如图 11-4 所示。

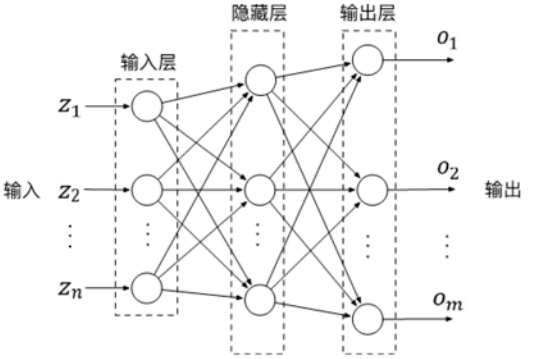


图 11-4　一个简单的神经网络作为生成模型的示例

生成模型与判别模型类似，也是由输入层、隐藏层、输出层组成的三层神经网络，不同的是，生成模型神经网络输入的是n维服从某一已知概率分布的随机数，如服从均匀分布或正态分布的随机噪声；输出为生成样本。采用记号描述生成模型对应的神经网络：G_L表示神经网络的层数，G_n^l表示第l层神经元的数量，$\mathrm{G}_\sigma_l(\cdot)$表示第$l$层神经元的激活函数，$\mathbf{G_W^l} \in \mathbb{R}^{n^{l-1}\times n^l}$表示第$l-1$层到第$l$层神经元的权值矩阵，$G_b^l \in \mathbb{R}^{n^l}$表示第$l$层神经元的偏置量，$G_z^l \in \mathbb{R}^{n^l}$表示第$l$层神经元的输入，$G_a^l \in \mathbb{R}^{n^l}$表示第$l$层神经元的输出。

3. 损失函数

判别模型和生成模型都有其各自的损失函数。判别模型的目标是准确地将输入的真实样本标记为真，输入的生成样本标记为假。因此，在判别模型存在两种损失：将输入的真实样本标记为假以及将输入的生成样本标记为真的损失。其损失函数可定义如下：

$$\mathrm{loss}_D = \mathrm{loss}_D^{\mathrm{real}} + \mathrm{loss}_D^{\mathrm{fake}}$$

其中，$\mathrm{loss}_D^{\mathrm{real}}$表述输入为真实样本时判别模型的损失，$\mathrm{loss}_D^{\mathrm{fake}}$表示输入为生成样本时判别模型的损失：

$$\text{loss}_D^{\text{real}} = -\frac{1}{N_\text{real}}\sum_{i=1}^{N_\text{real}}[y^{(i)}\log D_a^{L(i)} + (1-y^{(i)})(1-\log D_a^{L(i)})]$$

$$\text{loss}_D^{\text{fake}} = -\frac{1}{N_\text{fake}}\sum_{i=1}^{N_\text{fake}}[y^{(i)}\log D_a^{L(i)} + (1-y^{(i)})(1-\log D_a^{L(i)})]$$

其中，N_real表示输出输入判别模型的真实样本数量，N_fake表示输入判别模型的生成样本数量，$N_\text{real} = N_\text{fake}$。$y^{(i)}$表示样本$i$输入判别模型时的期望输出：

$$y^{(i)} = \begin{cases} 1 & i\text{为真实样本} \\ 0 & i\text{为生成样本} \end{cases}$$

因此，判别模型的损失函数可简化为：

$$\text{loss}_D = \text{loss}_D^{\text{real}} + \text{loss}_D^{\text{fake}}$$

$$= -\frac{1}{N_\text{real}}\sum_{i=1}^{N_\text{real}}[\log D_a^{L(i)}] - \frac{1}{N_\text{fake}}\sum_{i=1}^{N_\text{fake}}[1-\log D_a^{L(i)}]$$

上面讨论了判别模型的损失函数，而生成模型的目标是能够生成欺骗判别模型的样本，因此损失函数可以定义为：

$$\text{loss}_G = -\frac{1}{N_\text{fake}}\sum_{i=1}^{N_\text{fake}}[y^{(i)}\log D_a^{L(i)} + (1-y^{(i)})(1-\log D_a^{L(i)})]$$

其中，N_fake为输入判别模型的生成样本数量。$y^{(i)}$表示输入为生成样本时，判别模型的期望输出，此处$y^{(i)} = 1$。因此，生成模型的损失函数可简化为：

$$\text{loss}_G = -\frac{1}{N_\text{fake}}\sum_{i=1}^{N_\text{fake}}[\log D_a^{L(i)}]$$

11.2.2　实例：基于朴素生成对抗网络生成手写体数字

以生成手写体数字图像为例，采用 MNIST 作为训练集，在 MNIST 训练集中，一张手写体图像大小为 28×28，因此依次设计判别模型以及生成模型。

1. 判别模型的设计

判别模型的神经网络由输入层、一个隐藏层、输出层组成，因此$D_L = 3$。

第一层，输入层：由 784 个神经元组成，$D_n^1 = 784$。将 28×28 的手写体图像按像素展

开为一维列向量$\boldsymbol{D_z}^1=\begin{bmatrix}\mathrm{img}_1\\ \vdots\\ \mathrm{img}_{784}\end{bmatrix}$，展开后的列向量$\boldsymbol{D_z}^1$作为输入传入判别模型。

第二层，隐藏层：由 128 个神经元组成，$\mathrm{D_}n^2=128$。输入层和第二层之间采用全连接，用$\boldsymbol{w}_{i,j}^2$表示输入层第i个神经元到第二层第j个神经元的权值，用b_j^2表示第二层第j个神经元的偏置量，因此$\boldsymbol{D_z}^2=[\boldsymbol{D_W}^2]^{\mathrm{T}}\cdot\boldsymbol{D_a}^1+\boldsymbol{D_b}^2=\boldsymbol{D_z}^1\cdot\boldsymbol{D_W}^2+\boldsymbol{D_b}^2$，其中$D_\boldsymbol{W}^2=\begin{bmatrix}\boldsymbol{w}_{1,1}^2 & \cdots & \boldsymbol{w}_{1,128}^2\\ \vdots & \ddots & \vdots\\ \boldsymbol{w}_{784,1}^2 & \cdots & \boldsymbol{w}_{784,128}^2\end{bmatrix}$，$\boldsymbol{D_b}^2=\begin{bmatrix}b_1^2\\ \vdots\\ b_{128}^2\end{bmatrix}$。第二层采用 ReLU 作为激活函数，因此$\boldsymbol{D_a}^2=\boldsymbol{D_\sigma}_2(D_z^2)=\mathrm{ReLU}(\boldsymbol{D_z}^2)$。

第三层，输出层：由 1 个神经元组成，$D_n^3=1$。第二层和输出层之间采用全连接，用$w_{i,j}^3$表示第二层第i个神经元到输出层第j个神经元的权值，用b_j^2表示输出层第j个神经元的偏置量，因此$\boldsymbol{D_z}^3=[\boldsymbol{D_W}^3]^{\mathrm{T}}\cdot\boldsymbol{D_a}^2+\boldsymbol{D_b}^3$，其中$\boldsymbol{D_W}^3=\begin{bmatrix}\boldsymbol{w}_{1,1}^3\\ \vdots\\ \boldsymbol{w}_{1,128}^2\end{bmatrix}$，$\boldsymbol{D_b}^3=[b_1^3]$。输出层采用 Sigmoid 作为激活函数，因此$\boldsymbol{D_a}^3=\boldsymbol{D_\sigma}_3(\boldsymbol{D_z}^3)=\mathrm{sigmoid}(\boldsymbol{D_z}^3)$，输出层输出结果即为$\boldsymbol{D_a}^3$。

2. 生成模型的设计

生成模型的神经网络由输入层、一个隐藏层、输出层组成，因此$G_L=3$。

第一层，输入层：由 100 个神经元组成，$G_n^1=100$。生成 100 个在–1 到 1 之间均匀分布的随机变量$\boldsymbol{G_z}^1=\begin{bmatrix}z_1\\ \vdots\\ z_{100}\end{bmatrix}$。

第二层，隐藏层：由 128 个神经元组成，$G_n^2=128$。输入层和隐藏层之间采用全连接，用$\boldsymbol{w}_{i,j}^2$表示输入层第i个神经元到第二层第j个神经元的权值，用b_j^2表示第二层第j个神经元的偏置量，因此$\boldsymbol{G_z}^2=[\boldsymbol{G_W}^2]^{\mathrm{T}}\cdot\boldsymbol{G_a}^1+\boldsymbol{G_b}^2=[\boldsymbol{G_W}^2]^{\mathrm{T}}\cdot\boldsymbol{G_z}^1+\boldsymbol{G_b}^2$，其中，$\boldsymbol{G_W}^2=\begin{bmatrix}\boldsymbol{w}_{1,1}^2 & \cdots & \boldsymbol{w}_{1,128}^2\\ \vdots & \ddots & \vdots\\ \boldsymbol{w}_{100,1}^2 & \cdots & \boldsymbol{w}_{100,128}^2\end{bmatrix}$，$\boldsymbol{G_b}^2=\begin{bmatrix}b_1^2\\ \vdots\\ b_{128}^2\end{bmatrix}$。第二层采用 ReLU 作为激活函数，因此$G_a^2=\boldsymbol{G_\sigma}_2(\boldsymbol{G_z}^2)=\mathrm{ReLU}(\boldsymbol{G_z}^2)$。

第三层，输出层：由 784 个神经元组成，$G_n^3=784$。输入层和第二层之间采用全连接，用$w_{i,j}^3$表示第二层第i个神经元到输出层第j个神经元的权值，用b_j^3表示输出层第j个神经元的偏置量，因此$\boldsymbol{G_z}^3=[\boldsymbol{G_W}^3]^T\cdot\boldsymbol{G_a}^2+\boldsymbol{G_b}^3$，其中$\boldsymbol{G_W}^3=\begin{bmatrix}w_{1,1}^3 & \cdots & w_{1,784}^3\\ \vdots & \ddots & \vdots\\ w_{128,1}^3 & \cdots & w_{128,784}^3\end{bmatrix}$，$\boldsymbol{G_b}^3=$

$\begin{bmatrix} b_1^3 \\ \vdots \\ b_{784}^3 \end{bmatrix}$。输出层采用 Sigmoid 函数将输出归一化到（0,1），因此$\boldsymbol{G_a}^3 = \boldsymbol{G_\sigma}_3(\boldsymbol{G_z}^3) =$ sigmoid$(\boldsymbol{G_z}^3)$。

上述过程完成了生成模型与判别模型的设计，当设计完毕之后，则对模型进行训练，训练过程如下。

① 加载真实样本：因为生成对抗网络中 MNIST 不会用到测试数据集，所以可以将训练数据集和测试数据集合二为一，作为生成对抗网络的真实样本，共计 70000 个真实样本。此处需要预处理真实样本，将手写体图像的像素值调整为 0 到 1 之间。这是由于生成模型输出层采用 Sigmoid 作为激活函数，因此生成模型产生的生成样本图像的像素值在 0 到 1 之间。

② 训练判别模型：以 64 为批量大小，将真实样本和生成样本分别输入判别模型，采用 Adam 优化方法训练判别模型，Adam 方法的默认学习速率设置为 0.002。

③ 训练生成模型：以 64 为批量大小，将服从（–1,1）均匀分布的随机变量输入生成模型，同样采用 Adam 优化方法训练生成模型，默认学习速率同样设置为 0.002。

④ 重复上述②③两步，将全部真实样本迭代训练多次，直至平衡。

图 11-5 中的左图为初始随机化值，右图为上述过程迭代 25 次后的结果。

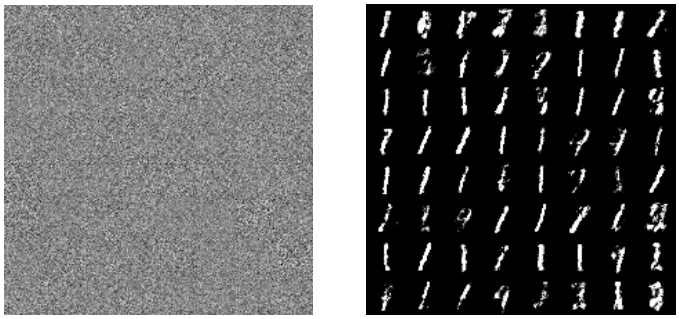


图 11-5　基于朴素生成对抗网络生成手写体数字示例

从图 11-5 中可以看出，已经能够模模糊糊看到生成的数字“1”。值得说明的是，在生成对抗网络中存在生成模型崩塌的问题，该问题会导致生成模型仅捕捉到真实样本的部分特征，只能生成相似的样本，新样本种类匮乏。

11.3 深度卷积生成对抗网络

11.3.1 产生背景

朴素生成对抗网络通过对抗训练能够较好地学习到真实样本的特征，而不必依赖特定的损失函数。然而，朴素生成对抗网络的训练过程十分不稳定，生成模型较大概率会生成无意义的输出，如模式崩塌时生成单一特征的样本。基于此，Alec Radford 等人将深度卷积引入朴素生成对抗网络，同时找到一组较好的网络拓扑结构，形成深度卷积生成对抗网络（Deep Convolution Generative Adversarial Networks，DCGAN）。自带深度卷积的生成对抗网络提出后，DCGAN 就广泛应用于图像处理相关领域。

一方面，在图像应用领域，理论和实践都已经证明：深度卷积神经网络是目前处理图像的最有效手段，广泛应用于图像分类、图像理解等。另一方面，生成对抗网络中的判别模型可以理解为一个二元分类器。

鉴于上述两点，DCGAN 将深度卷积神经网络引入到判别模型。判别模型将输入的图像信息经过深度卷积神经网络后，提取图像特征，逐层减小图像尺寸，进而输出原始图像信息的抽象表达，最后达到图像分类的目的。

DCGAN 中生成模型的处理流程可近似看作判别模型的逆向过程，其目标是生成图像，将一组特征值逐层恢复成图像。生成模型将输入的一维随机变量，经过深度反卷积神经网络（可理解为深度卷积神经网络的逆向过程），通过上采样，逐层放大原始信息的特征，最终排列成新的图像，生成新的样本。

11.3.2 模型改进

除了在朴素生成对抗网络中引入卷积层外，深度卷积生成对抗网络对模型结构的优化点还包括增加正则化、取代池化层、选择特定激活函数等。

（1）增加正则化。在判别模型和生成模型中都采用正则化，这样可以防止生成模型将所有新生成的样本都收敛到同一个点，造成生成样本多样性的匮乏；还可以将梯度传播到每一层，加快训练。但是正则化方法不能应用到判别模型的输入层和生成模型的输出层，这是因为如果将正则化应用到判别模型和生成模型的所有层会影响模型的稳定性。

（2）取代池化层。在判别模型中使用有步长的卷积核代替池化层，学习降采样；在生成模型中使用反卷积，学习升采样。这是因为卷积神经网络中池化层的目的主要是为了加速训练，而 DCGAN 使用了正则化和带步长的卷积核，训练速度已经有所提升，因此池化层没有存在的必要。

（3）选择特定的激活函数。判别模型的所有层都使用 Leaky ReLU 作为激活函数；生成模型除输出层外都使用 ReLU 作为激活函数，而输出层采用 Tanh 作为激活函数。

作个形象类比，判别模型可比作图像理解，生成模型可比作图像绘画。图像理解的流程是：逐层剥离细节信息，最终获取图像的基本特征。而图像绘画的流程则是：构思结构、勾画轮廓、绘制细节、填充色彩，逐步丰富细节信息。

11.3.3　网络结构

深度卷积生成对抗网络和朴素生成对抗网络一样，也包括判别模型和生成模型，其中目标函数同朴素生成对抗网络，此处不再介绍。

1. 判别模型

深度卷积生成对抗网络中的判别模型如图 11-6 所示。

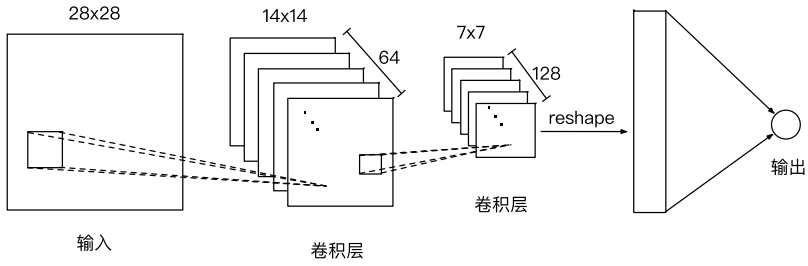


图 11−6　深度卷积生成对抗网络中的判别模型

判别模型是由输入层、两个隐藏层和输出层组成的四层神经网络，其中两个隐藏层都是卷积层。与朴素 GAN 相似，输入 DCGAN 判别模型的是真实样本或生成样本，输出的是当前样本为真实样本而非生成样本的概率。采用以下记号描述 DCGAN 判别模型对应的神经网络：D_L表示神经网络的层数；$D_\sigma_l(\cdot)$表示第l层神经元的激活函数；$\boldsymbol{D_W}^l \in \mathbb{R}^{n^{l-1}\times n^l}$表示第$l-1$层到第$l$层神经元的权值矩阵（卷积核）；$D_b^l \in \mathbb{R}^{n^l}$表示第$l$层神经元的偏置量；$D_z^l \in \mathbb{R}^{n^l}$表示第$l$层神经元的输入；$D_a^l \in \mathbb{R}^{n^l}$表示第$l$层神经元的输出；$D$_norm表示正则化函数；$D$_conv表示卷积函数；$D$_strides表示卷积步长。

2. 生成模型

深度卷积生成对抗网络中的生成模型如图 11-7 所示。

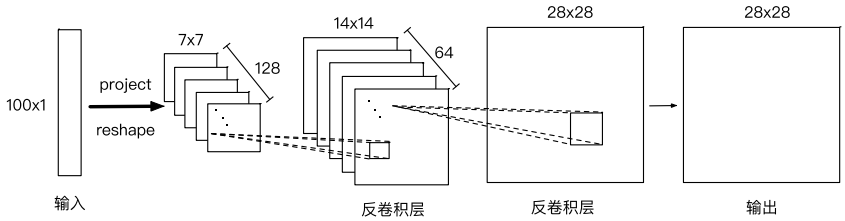


图 11-7　深度卷积生成对抗网络中的生成模型

生成模型与判别模型类似，也是由输入层、两个隐藏层和输出层组成的四层神经网络，其中两个隐藏层都是反卷积层。不同于判别模型的是，生成模型神经网络输入的是服从某一已知概率分布的随机数，如服从均匀分布或正态分布的随机噪声；输出为生成样本。采用以下记号描述生成模型对应的神经网络：G_L表示神经网络的层数；$G_\sigma_l(\cdot)$表示第l层神经元的激活函数；$\boldsymbol{G_W^l} \in \mathbb{R}^{n^{l-1} \times n^l}$表示第$l-1$层到第$l$层神经元的权值矩阵（卷积核）；$G_b^l \in \mathbb{R}^{n^l}$表示第$l$层神经元的偏置量；$\boldsymbol{G_z^l} \in \mathbb{R}^{n^l}$表示第$l$层神经元的输入；$\boldsymbol{G_a^l} \in \mathbb{R}^{n^l}$表示第$l$层神经元的输出；$G_\boldsymbol{norm}$表示正则化函数；$G_\boldsymbol{deconv}$表示反卷积函数；$G_\boldsymbol{strides}$表示卷积步长。

11.3.4　实例：基于深度卷积对抗网络生成手写体数字

依然以手写体图片训练集 MNIST 为例，训练集中手写体图片大小为 28×28，分别设计深度卷积对抗网络的判别模型和生成模型。

1. 判别模型设计

判别模型的神经网络由输入层、两个隐藏层和输出层组成，因此$D_L = 4$。

第一层，输入层。输入层的输入可表示为

$$\boldsymbol{D_z^1} = [\text{img_height}, \text{img_width}, \text{img_color_dim}]$$

由于手写体图像为 28×28 大小的灰度图，即单通道图像，因此$\text{img_heigt} = 28$，$\text{img_width} = 28$，$\text{img_color_dim} = 1$。

第二层，隐藏层。为卷积层，卷积核为：

$$\boldsymbol{D_W^2} = [5, 5, \text{img_color_dim}, 64]$$

其中卷积核的大小为 5×5，卷积核深度为 64，初始参数为服从方差为 0.02 的正态分布的随机变量。偏置量为：

$$\boldsymbol{D_b^2} = \begin{bmatrix} b_1^2 \\ \vdots \\ b_{64}^2 \end{bmatrix}$$

偏置量的初始值为 0.0。卷积层的输入$\boldsymbol{D_z}^2$可通过如下公式计算：

$$\boldsymbol{D_z}^2 = D_\text{conv}(\boldsymbol{D_a}^1, \boldsymbol{D_W}^2, \text{D_strides}) + D_b^2$$

其中 D_$strides = [1,2,2,1]$，即卷积步长为 2，且D_conv函数采用零填充，确保经过卷积计算后输出图像的大小变为输入图像大小的一半，此时输出的图像大小变为 14×14。再经过正则化和激活函数产生卷积层的输出：

$$\boldsymbol{D_a}^2 = \boldsymbol{D_\sigma}_2(D_\text{norm}(\boldsymbol{D_z}^2)) = \text{LReLU}(D_\text{norm}(\boldsymbol{D_z}^2))$$

此处激活函数选取 Leakly ReLU，$\boldsymbol{D_a}^2$的维度是$[14, 14, 64]$。

第三层，隐藏层。为卷积层，卷积核为：

$$\boldsymbol{D_W}^3 = [5, 5, 64, 128]$$

其中卷积核的大小为 5×5，卷积核深度为 128，初始参数为服从方差为 0.02 的正态分布的随机变量。偏置量为：

$$\boldsymbol{D_b}^3 = \begin{bmatrix} b_1^3 \\ \vdots \\ b_{128}^3 \end{bmatrix}$$

偏置量的初始值为 0.0。卷积层的输入$\boldsymbol{D_z}^3$可通过如下公式计算：

$$\boldsymbol{D_z}^3 = D_\text{conv}(\boldsymbol{D_a}^2, \boldsymbol{D_W}^3, \text{D_strides}) + \boldsymbol{D_b}^3$$

其中 D_$strides = [1,2,2,1]$，即卷积步长为 2，且D_conv函数采用零填充，确保经过卷积计算后输出图像的大小变为输入图像大小的一半，此时输出的图像大小变为 7×7。再经过正则化和激活函数产生卷积层的输出：

$$\boldsymbol{D_a}^3 = \boldsymbol{D_\sigma}_3(D_\text{norm}(\boldsymbol{D_z}^3)) = \text{LReLU}(D_\text{norm}(\boldsymbol{D_z}^3))$$

此处激活函数选取 Leakly ReLU，$\boldsymbol{D_a}^3$的维度是$[7, 7, 128]$。接着将$\boldsymbol{D_a}^3$的维度调整为6272的列向量，其中$6272 = 7 \times 7 \times 128$。

第四层，输出层。输出层有一个节点，与二分类判别器类似，输出值在 0~1 之间，用以表征输入样本属于真实样本而非生成样本的概率。第三层和输出层之间采用全连接，用$\boldsymbol{w}_{i,j}^4$表示第三层第i个神经元到输出层第j个神经元的权值，用$\boldsymbol{b}_j^4$表示输出层第j个神经元的偏置量，因此$\boldsymbol{D_z}^4 = [\boldsymbol{D_W}^4]^\text{T} \cdot \boldsymbol{D_a}^3 + \boldsymbol{D_b}^4$，其中$\boldsymbol{D_W}^4 = \begin{bmatrix} \boldsymbol{w}_{1,1}^4 \\ \vdots \\ \boldsymbol{w}_{6272,1}^4 \end{bmatrix}$，$\boldsymbol{D_b}^4 = [\boldsymbol{b}_1^4]$。$\boldsymbol{D_a}^4 = \boldsymbol{D_z}^4$直接作为输出层输出结果。

2. 生成模型设计

判别模型的神经网络由输入层、两个隐藏层和输出层组成，因此$G_L = 4$。

第一层，输入层。由 100 个神经元组成。生成 100 个在-1 到 1 之间均匀分布的随机变量，$\boldsymbol{G_z}^1 = \begin{bmatrix} z_1 \\ \vdots \\ z_{100} \end{bmatrix}$作为生成模型的输入。

第二层，隐藏层。为全连接和维度转换层，由 6272 个神经元组成。输入层和隐藏层之间采用全连接，用$\boldsymbol{w}_{i,j}^2$表示输入层第i个神经元到隐藏层第j个神经元的权值，用b_j^2表示隐藏层第j个神经元的偏置量，因此$\boldsymbol{G_z}^2 = [\boldsymbol{G_W}^2]^{\mathrm{T}} \cdot \boldsymbol{G_a}^1 + \boldsymbol{G_b}^2 = [\boldsymbol{G_W}^2]^{\mathrm{T}} \cdot \boldsymbol{G_z}^1 + \boldsymbol{G_b}^2$，其中$\boldsymbol{G_W}^2 = \begin{bmatrix} \boldsymbol{w}_{1,1}^2 & \cdots & \boldsymbol{w}_{1,6272}^2 \\ \vdots & \ddots & \vdots \\ \boldsymbol{w}_{100,1}^2 & \cdots & \boldsymbol{w}_{100,6272}^2 \end{bmatrix}$，$\boldsymbol{G_b}^2 = \begin{bmatrix} \boldsymbol{b}_1^2 \\ \vdots \\ \boldsymbol{b}_{6272}^2 \end{bmatrix}$。第二层的输入数据再经过正则化和激活函数，此层的输出结果为：

$$\boldsymbol{G_a}^2 = \boldsymbol{G_\sigma}_2(G_\mathrm{norm}(\boldsymbol{G_z}^2)) = \mathrm{ReLU}(D_\mathrm{norm}(\boldsymbol{G_z}^2))$$

此处激活函数选取 ReLU，$\boldsymbol{G_a}^2$的维度是[6272]。接着将$\boldsymbol{D_a}^2$的维度调整为[7,7,128]。

第三层，隐藏层，为反卷积层。卷积核为：

$$\boldsymbol{G_W}^3 = [5, 5, 64, 128]$$

其中卷积核的大小为 5×5，卷积核深度为 64，初始参数为服从方差为 0.02 的正态分布的随机变量。偏置量为：

$$\boldsymbol{G_b}^3 = \begin{bmatrix} b_1^3 \\ \vdots \\ b_{64}^3 \end{bmatrix}$$

偏置量的初始值为 0.0。反卷积层的输入$\boldsymbol{G_z}^3$可通过如下公式计算：

$$\boldsymbol{G_z}^3 = G_\mathrm{deconv}(\boldsymbol{G_a}^2, \boldsymbol{G_W}^3, G_\mathrm{strides}) + \boldsymbol{G_b}^3$$

其中$G_\mathrm{strides} = [1,2,2,1]$，即反卷积步长为 2。此处通过上采样，经过反卷积计算后输出图像的大小变为输入图像大小的 2 倍，此时输出的图像大小变为 14×14。再经过正则化和激活函数产生反卷积层的输出：

$$\boldsymbol{G_a}^3 = \boldsymbol{G_\sigma}_3(G_\mathrm{norm}(\boldsymbol{G_z}^3)) = \mathrm{ReLU}(G_\mathrm{norm}(\boldsymbol{G_z}^3))$$

此处激活函数选取 ReLU，$\boldsymbol{G_a}^3$的维度是$[14, 14, 64]$。

第四层，隐藏层。为反卷积层。卷积核为：

$$\boldsymbol{G_W}^4 = [5,5,1,64]$$

其中卷积核的大小为 5×5，卷积核深度为 1，初始参数是服从方差为 0.02 的正态分布的随机变量。偏置量为：

$$G_b^4 = [b_1^4]$$

偏置量的初始值为 0.0。反卷积层的输入$\boldsymbol{G_z}^4$可通过如下公式计算：

$$\boldsymbol{G_z}^4 = G_\text{deconv}(\boldsymbol{G_a}^3, \boldsymbol{G_W}^4, G_\text{strides}) + G_b^4$$

其中$G_\text{strides} = [1,2,2,1]$，即反卷积步长为 2。此处通过上采样，经过反卷积计算后输出图像的大小变为输入图像大小的 2 倍，此时输出的图像大小变为 28×28，即真实样本图像大小。再经过激活函数产生反卷积层的输出：

$$\boldsymbol{G_a}^4 = \boldsymbol{G_\sigma}_4(\boldsymbol{G_z}^3) = \text{Tanh}(\boldsymbol{G_z}^3)$$

此处激活函数选取 Tanh，$\boldsymbol{G_a}^4$的维度是$[28,28,1]$，且输出图像的像素值在–1 到 1 之间。

基于上述设计的生成模型和判别模型，其训练过程可以按照如下步骤进行。

❶ 加载真实样本。因为生成对抗网络中 Mnist 不会用到测试数据集，所以可以将训练数据集与测试数据集合二为一，作为生成对抗网络的真实样本，共计 70000 个真实样本。此处需要预处理真实样本，将手写体图像的像素值调整到–1 到 1 之间。这是由于生成模型输出层采用 Tanh 作为激活函数，因此生成模型产生的生成样本图像的像素值在–1 到 1 之间。

❷ 训练判别模型。以 64 为批量大小，将真实样本和生成样本分别输入判别模型，采用 Adam 优化方法训练判别模型，Adam 方法的默认学习速率设置为 0.002。

❸ 训练生成模型。以 64 为批量大小，将服从(–1,1)均匀分布的一维随机变量输入生成模型，同样用 Adam 优化方法训练生成模型，Adam 方法的默认学习速率同样设置为 0.002。

❹ 重复❷❸两步，将全部真实样本迭代训练直至平衡。

图 11-8 中的左图为初始随机化值，右图为上述训练过程迭代 25 次后的结果。

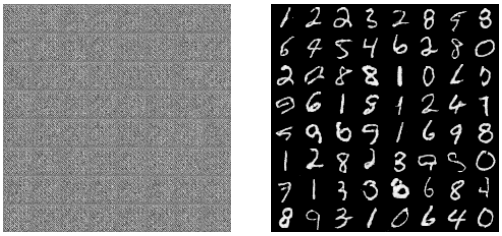


图 11-8　基于深度卷积对抗网络生成的手写数字示例

从图 11-8 中可以看出，对比朴素生成对抗网络，深度卷积生成对抗网络在手写体图像生成的应用中能够生成较高质量的图像。

11.4 条件生成对抗网络

朴素生成对抗网络中的生成模型通过将服从均匀分布的随机变量映射到新的样本，这在理论上可以生成和真实样本一模一样的新样本。但是，不同于传统生成模型的预建模方式，朴素生成对抗网络生成新样本的方式太过自由，从而导致生成过程不太受控。为了解决上述问题，Mehdu Mirza 等学者提出条件生成对抗网络（Conditional Generative Adversarial Networks，CGAN），即通过同时给生成模型和判别模型增加约束条件，让生成的样本符合预期，进而引导生成模型生成新样本的过程。条件生成对抗网络的一个优势在于可以对多形式的数据生成提供更好的表征。

11.4.1 网络结构

条件生成对抗网络的基本结构如图 11-9 所示。

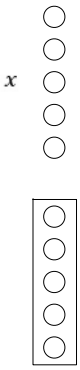


图 11-9 条件生成对抗网络的结构示例

从图 11-9 中可以看出，条件生成对抗网络是对朴素生成对抗网络的扩展。在条件生成对抗网络中，额外信息会作为条件信息被同时增加到生成模型和判别模型，前一层输出原始信息都会联合约束信息，从而形成新的信息，逐层传递。其中，额外增加的信息可以是任意辅助信息，如样本标注或其他模态的信息。类似于朴素生成对抗网络，条件生成对抗网络可以理解为携带条件概率的零和博弈问题，可以用数学语言描述如下：

$$\min_G \max_D V(D,G) = E_{x\sim p_{\text{data}}(x)}[\log D(x|y)] + E_{z\sim p_z(z)}[\log(1 - D(G(z|y)))]$$

其中，y就是新增的条件信息。直观上理解，假设有两个事件记为 A 和 B，当 B 发生时 A

发生的概率大于等于事件 A 发生的概率，因此学习条件概率的难度较低，这就是提出条件生成对抗网络的原因。

11.4.2　实例：CGAN 结合 DCGAN 生成手写体数字

依然以手写体训练集 MNIST 为例，在 DCGAN 的基础上增加图像标注作为约束信息实现 CGAN。判别模型和生成模型与 DCGAN 基本类似，唯一区别在于输入 CGAN 的数据有三类：真实样本、真实样本的标注和随机变量。CGAN 会在判别模型和生成模型的层与层连接之间增加约束信息合成层，带约束条件的深度卷积生成对抗网络的判别模型结构和生成模型结构分别如图 11-10 和图 11-11 所示。

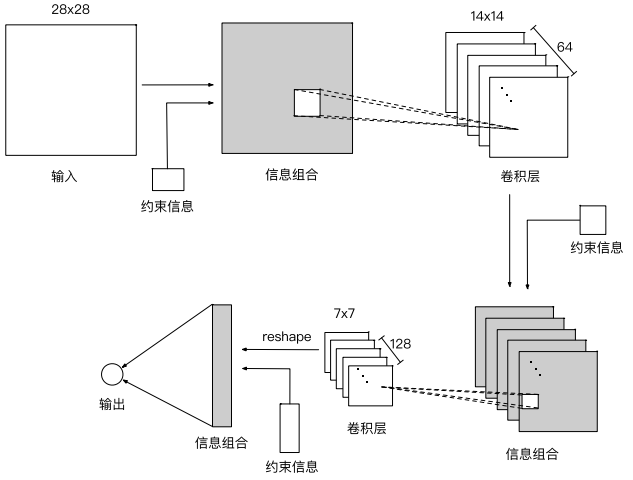


图 11-10　带约束条件的深度卷积生成对抗网络的判别模型结构

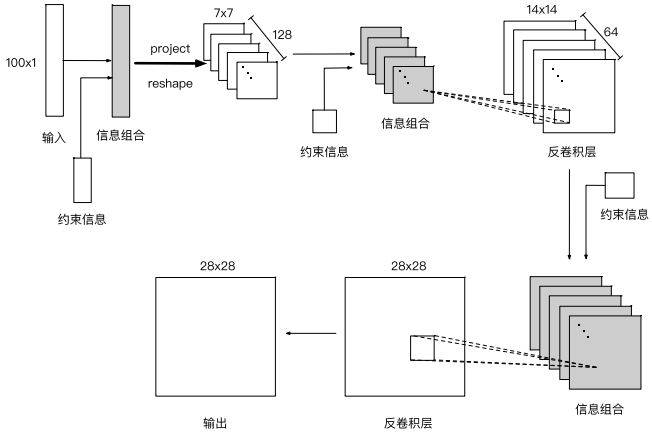


图 11-11　带约束条件的深度卷积生成对抗网络的生成模型结构

有一点需要注意，以生成模型输入层为例，z是服从均匀分布的一维随机变量，为连续变量；而作为约束信息的标注却是离散特征，即手写体标注。为了解决连续变量和离散特征的联合问题，为此，需要对离散变量进行独热编码。

独热编码，即使用N位状态寄存器对N个状态进行编码，每个状态都有其独立的寄存器位，并且在同一时刻有且仅有一个寄存器位有效。用$f(x)$表示独热编码过程，以手写体图像为例。将手写体图像的标注看作特征A，该特征共有 10 种可能值：0、1、2、3、4、5、6、7、8、9。经过独热编码后，特征A会变成 10 个二元特征，如$f(1)=01000000000$，并且这 10 个二元特征互斥，每次仅有一个处于激活状态。离散特征进行独热编码后，每一维度的特征都可以看成是连续的特征。将离散特征经过独热编码后，会使特征之间的距离计算更加合理。下面以欧氏空间相似度计算为例来说明，用$d(x,y)$代表欧氏空间的相似度。若不采用独热编码，则$d(0,5)=5$、$d(0,9)=9$，说明 0 和 9 之间的相似度低于 0 和 5 的相似度，这显然是不合理的，因为此处 0、5、9 仅仅是一种标注，它们之间的相似度应该一致。采用独热编码：

$$d(f(0),f(5))=d(1000000000,0000100000)=\sqrt{2}$$

$$d(f(0),f(9))=d(1000000000,0000000001)=\sqrt{2}$$

经过独热编码后，不同标注之间的距离变得一致，更加合理。

基于带约束条件的深度卷积生成对抗网络生成的手写体数字图像如图 11-12 所示，其中，左图为 1 次迭代后生成的图像，右图为 25 次迭代后生成的图像。从图 11-12 可以看到，结合 CGAN 的 DCGAN 已经能够根据约束信息生成相对应的、非常明晰的手写体数字图像。

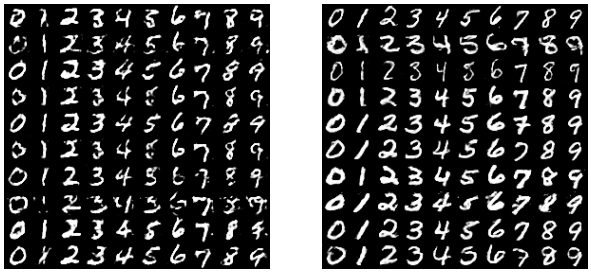


图 11-12　CGAN 结合 DCGAN 生成手写体数字图像的示例

11.5　瓦瑟斯坦生成对抗网络

11.5.1　概述

自生成对抗网络提出之日起，模型训练困难、生成样本多样性匮乏、损失函数无法指示训

练进展等问题就与之相伴。训练 GAN 不仅需要精心设计模型的超参，而且需要控制判别模型和生成模型的迭代训练，防止判别模型过于准确而导致生成模型无法进一步训练。

生成对抗网络的优化目标可以理解为通过最小化生成样本和真实样本之间的概率分布差异，来逼近判别模型和生成模型的纳什均衡。朴素生成对抗网络中，当判别模型近似最优时，目标函数$\min_G \max_D V(D,G)$ 可解释为最小化生成样本与真实样本概率分布之间的 JS 距离，其中 JS 距离是信息论中一种用于衡量两个概率分布相似度的指标。当两个概率分布的样本空间几乎不重叠时，它们之间的 JS 距离会退化为常量，从而造成生成模型梯度为零，这就是朴素 GAN 梯度消失的原因。为了解决上述几个问题，Martin Arjovsky 引入瓦瑟斯坦（Wasserstein）距离来衡量两个概率分布之间的距离，并提出瓦瑟斯坦生成对抗网络（Wasserstein GAN, WGAN）。瓦瑟斯坦距离定义如下：

$$W(P_r,P_g)=\inf_{\gamma\sim\Pi(P_r,P_g)}E_{(x,y)\sim\gamma}[\|x-y\|]$$

其中，P_r表示真实样本的概率分布，P_g表示生成样本的概率分布，$\Pi(P_r,P_g)$表示P_r和P_g组合后所有可能的联合概率分布集合。对于每一个可能的联合分布γ，可以通过采样获得真实样本x和生成样本y，并计算出样本距离的期望$E_{(x,y)\sim\gamma}[\|x-y\|]$。而瓦瑟斯坦距离就是在所有联合分布中距离期望能够取到的下确界。

相比于 JS 距离，即使生成样本和真实样本的概率分布没有重叠，瓦瑟斯坦依旧具备衡量两个分布间距离的能力，因此理论上可以解决训练过程中梯度消失的问题。

11.5.2　差异化

瓦瑟斯坦生成对抗网络的判别模型和生成模型的设计与朴素生成对抗网络或深度卷积生成对抗网络相似，其主要区别在于以下四个方面。

（1）判别模型输出层的激活函数方面。

瓦瑟斯坦生成对抗网络使用瓦瑟斯坦距离衡量生成样本和真实样本概率分布之间的距离，因此输出层无须经过 Sigmoid 激活函数。

（2）损失函数方面。

瓦瑟斯坦生成对抗网络中判别模型的损失函数定义为：

$$\text{loss}_D=\frac{1}{N_\text{fake}}\sum_{i=1}^{N_\text{fake}}[D_a^{L(i)}]-\frac{1}{N_\text{real}}\sum_{i=1}^{N_\text{real}}[D_a^{L(i)}]$$

瓦瑟斯坦生成对抗网络中生成模型的损失函数定义为：

$$\text{loss}_G = -\frac{1}{N_\text{fake}} \sum_{i=1}^{N_\text{fake}} [D_a^{L(i)}]$$

其中，N_fake表示输入判别模型的生成样本数量，N_real表示输入判别模型的真实样本数量，$N_\text{fake} = N_\text{real}$。

（3）梯度下降的优化算法方面。

瓦瑟斯坦生成对抗网络中使用 RMSProp 优化算法，而在朴素生成对抗网络中则采用 Adam 优化算法。

（4）训练过程方面。

在使用梯度下降算法更新判别模型训练参数后，需要将判别模型的参数的绝对值截断到一个固定常数范围内，如[-0.01,100]。

瓦瑟斯坦生成对抗网络的贡献有以下几方面。

（1）解决了训练稳定性的问题。无须平衡判别模型和生成模型的训练程度，即使判别模型达到最优，生成模型也存在优化空间。

（2）基本解决生成模型崩塌问题，保证生成样本的多样性。

（3）引入瓦瑟斯坦距离来指示训练进展，瓦瑟斯坦距离越小说明训练得越好，生成的样本质量越高。

11.5.3 实例：WGAN 结合 DCGAN 生成手写体数字

WGAN 结合 DCGAN 生成手写体数字的判别模型设计、生成模型设计以及训练过程都与 DCGAN 相似，唯一区别在于损失函数的定义。鉴于前文已经详细阐述过 WGAN 的损失函数，此处不再赘述。

图 11-13 为使用瓦瑟斯坦距离衡量生成样本和真实样本概率分布距离的 DCGAN 生成的手写体数字图像，所示图为 25 次迭代后的生成图像。对比图 11-8 和图 11-13 可知，使用基于瓦瑟斯坦距离的损失函数后，生成的手写体图像质量略优于原始 DCGAN。这一方面体现了 WGAN 的优点，另一方面也说明了 DCGAN 自身就具有较好的网络拓扑结构。

图 11-13　WGAN 结合 DCGAN 生成手写体数字图像的示例

11.6　生成对抗网络的探索

11.6.1　价值与意义

人工智能从某种意义上讲可以分为两类：一类是让机器变得像人一样去思考问题，分析问题，进而解决问题；另一类是让机器自己拥有思考问题、分析问题、解决问题的能力，而不局限于人类的经验和智慧。

在生成对抗网络出现之前，我们研究生成模型的方法可归属为第一类，其一般思路是：首先根据经验假设真实样本符合某一分布，如正态分布；然后通过抽样拟合，计算假设分布的参数。上述生成模型的输出结果严重依赖第一步假设的分布，而假设的部分很大程度上取决于做出假设的人及其经验。

但是，面对浩瀚的宇宙，人类的经验往往十分匮乏。生成对抗网络归属于第二类，让机器自己理解数据、研究生成模型。生成对抗网络引入对抗的理念，用模型生成数据，再用另一个模型判别生成效果，如此循环迭代、反复修正两个模型，最终达到动态平衡，达到机器对真实样本的理解。

生成对抗网络将对抗机制引入机器学习领域，判别模型可看成是有监督学习，而生成模型可看成是无监督学习。通过判别模型和生成模型两个神经网络的对抗训练，能够有效地生成符合真实样本分布的新样本。由于采用神经网络作为判别模型和生成模型的结构，因此生成对抗网络还具备生成高维数据的能力。

在模型训练过程中，生成对抗网络通过反向传播训练模型参数，其训练过程既不依赖马尔科夫链的反复采样，也不需要近似推理。生成模型借助判别模型反向传递的梯度来训练、优化参数，而非直接使用真实样本数据进行参数调优。正如上述章节所介绍，生成对抗网络中判别模型D和生成模型G仅要求是可微函数，由此可见，任何一个可微函数都能够用于生成对抗网络。因此生成对抗网络可以和其他神经网络相结合，从而赋予生成对抗网络的多种优点。

11.6.2 面临的问题

生成对抗网络是目前深度学习中发展较为迅速的网络结构，通过生成对抗网络，也能够凸显出生成对抗网络的优势，但是目前生成对抗网络还面临诸多问题。

（1）可解释性差。生成对抗网络中生成模型可看作一个函数映射，输入的是随机变量，输出的是符合真实样本分布的新样本数据，但是由于无法显式地表达该概率分布，因此生成对抗网络难以解释和说明。

（2）训练难度大。生成对抗网络由判别模型和生成模型组成，训练过程中需要交替优化上述两个模型，Ian Goodfellow 在论文中提到需要循环优化判别模型k次，接着才优化生成模型 1 次，这主要是为了让判别模型保持一定的判别准确度。另一方面，判别模型训练得越好，生成模型的梯度消失就越严重。由此可见，参数k以及生成对抗网络中的其他超参数的设置都需要大量的实践，因此生成对抗网络训练比较困难。

（3）训练难以收敛。凸优化函数存在全局最优解，而生成对抗网络可看作判别模型和生成模型双方的博弈，其优化函数往往并非凸函数。目前，理论上还无法判断模型的收敛性，关于如何训练生成对抗网络以达到全局最优点尚未有相关研究结论。

（4）模式崩塌问题。生成对抗网络可建模为极小极大问题，判别模型和生成模型有各自的损失函数，因此在训练过程中难以判断当前网络处于的优化状态。在生成模型的训练过程中，生成模型可能会捕捉到判别模型的某些缺陷，从而大量甚至全部生成能欺骗判别模型的特征，导致生成模型退化，此时生成模型学习到的特征仅集中在全部特征的几个地方，生成的新样本在人类看来也几乎相似。可尝试通过小批量生成对抗网络（Minibatch GAN）来解决模式崩塌的问题，小批量生成对抗网络将真实数据分为多个批量，同时保证每个批量的样本足够多样。

（5）缺少科学的评估标准来衡量生成对抗网络效果。以生成图像为例，人类能够借助主观能动性，通过图像内容、清晰度、颜色等多维度判别一张图像的真伪，但是机器却无法量化和评价生成图像的质量。只有建立完善、公认、客观的衡量图像质量的标准，才能在此基础上科学优化地生成对抗网络。

11.6.3 应用场景示例

生成对抗网络可用于真实样本数据概率分布的建模，并生成与真实数据相同分布的新数据。生成对抗网络可以应用于图像与视觉领域，如提升图像分辨率、还原遮挡或破损图像、预测视频帧等，还可以应用于自然语言处理领域，如生成对话文本、基于文本描述生成图像等。除图像视觉和自然语言处理等计算机常见领域外，生成对抗网络还广泛应用于其他领域，如药物匹配、历史档案图像检索等。本节将通过图像分辨率提升、从边缘图重构物体这两个案例阐述生

成对抗网络的应用。

1. 提升图像分辨率

Twitter 利用生成对抗网络将一个低分辨率图像转换成具有丰富细节的高分辨率图像。在此应用中，生成模型是参数化的残差网络，其作用是：输入低分辨率图像，输出高分辨率图像。判别模型是 VGG 网络（由牛津大学 VGG 团队提出的深度卷积神经网络），其作用是：判别输入判别模型的是真实高分辨率图像还是生成模型生成的高分辨率图像。图 11-14 为基于生成对抗网络进行图像分辨率提升示例。

（a）模糊图

（b）SRGAN还原图　　（c）原始图

图 11-14　基于生成对抗网络提升图像分辨率示例

图 11-14 中（c）图为原始高分辨率图像，（a）图为对原始图像宽高按四倍采样生成的低分辨率图像，（b）图为采用生成对抗网络还原的高分辨率图像。从图中可以看到，生成对抗网络生成了具有丰富细节的高分辨率图像，人眼几乎无法区分还原图像和原始高分辨率图像。基于此项技术，Twitter 等视频网站可以将高分辨率视频转换为低分辨率视频在后端传输，在前端展示播放时再通过此技术将低分辨率视频还原成高分辨率视频。如此，能够在不降低用户观影体验的前提下，降低传输成本。

2. 从边缘图重构物体

此应用选取带约束条件的生成对抗网络，输入的是随机噪声和约束条件，其中约束条件即为物体的边缘轮廓图，输出的则是真实物体图像。如图 11-15 所示，左图是手工绘制的手提包轮廓，右图是生成对抗网络生成的真实手提包图像。此技术可以用于漫画领域，如给手绘漫画着色等，降低动漫创作成本。

图 11-15　基于生成对抗网络从边缘重构物体示例

11.6.4 未来探索

生成对抗网络的核心思想及其看待问题的角度，已经给机器学习领域注入了新的能量，也必定会对机器学习领域带来深远的影响。未来生成对抗网络将极有可能与下列领域碰撞出火花。

（1）虚拟现实（Virtual Reality，VR）。使用生成对抗网络实现 VR 场景和真实物体的 3D 建模，解决现阶段 3D 建模的成本高、难度大等难题。

（2）无人驾驶。将生成对抗网络用于无人驾驶中的无监督或半监督学习，利用生成对抗网络生成与真实交通场景一致的多种多样的道路情况，辅助无人驾驶模型的训练和优化。

（3）信息安全。设计生成对抗网络，并利用对抗学习的机制实现信息加密的自学习，在模型训练过程中不断修改加密算法，进而生成复杂的密码。

（4）训练样本自动标注。标注是一项耗时而枯燥的工作，但是准确标注过的训练样本却对模型的训练效果起着关键作用。为了解决上述问题，可以利用生成对抗网络，将不带标注的样本和标注通过合成的方式生成带有标注的样本。以图像标注为例，当需要在图像中标注出小狗时，传统做法是：人工使用标注工具，标注出小狗在图像中的位置。而借助生成对抗网络，可以将一张草坪图像和一张小狗图像输入生成对抗网络，接着生成对抗网络会合并两幅图像，从而生成带有小狗标注的符合自然现象的真实图像。

生成对抗网络为创造无监督学习模型提供了强有力的算法框架，未来生成对抗网络将会更多地应用于无监督学习领域。与此同时，生成对抗网络与特征学习、强化学习的结合研究，也是未来的发展趋势之一。生成对抗网络亟待广大读者和科研从业人员深入研究，正所谓前路漫漫，吾将上下而求索。

11.7 本章小结

本章介绍了生成对抗网络，不仅介绍了生成对抗网络的背景和发展，还介绍了生成对抗网络对抗训练的核心思想，同时详细介绍了四种生成对抗网络结构，全连接神经网络实现的朴素生成对抗网络、深度卷积生成对抗网络、条件生成对抗网络以及瓦瑟斯坦生成对抗网络。并以手写体图像生成为例，阐述了朴素生成对抗网络、深度卷积生成对抗网络和条件生成对抗网络的实现过程。

本章分析了现阶段生成对抗网络面临的问题和挑战，以及生成对抗网络的应用场景。最后，总结了生成对抗网络对机器学习领域的意义，同时对生成对抗网络的未来应用前景进行了展望和畅想。

第 12 章

深度强化学习

目前的学习深度一直是在计算机领域发展，语音识别、自然语言理解等领域已经取得了较好突破，相关技术逐渐成熟并开始影响我们的生活。近些年，深度强化学习成为了人工智能技术领域的热门研究方向之一，其理论的通用性在各个业务领域都能较好地适应，吸引了众多研究者。在传统的机器学习分类中，并不包含强化学习，但现在也常常划分为三类：分别是有监督学习、无监督学习以及强化学习。

12.1 概述

众多的机器学习研究者都在思考的一个最基本的问题是“人类如何学习新技能”。研究的原因是，倘若能够有效地解答这个问题，一方面人类就能掌握更快速的学习方法，以最短的时间获得最佳的学习效率；另一方面是根据人类学习的方法，使得机器也能掌握这样的学习方法。当机器同人类一样真正掌握学习的方法之后，也许能创造出真正的人工智能，但是目前绝大部分的机器学习都是以模仿人类行为为主，还处于模仿阶段。强化学习则是与环境相关的一种学习方法。

12.1.1 概要

强化学习（Reinforcement learning，RL）也被称作增强学习，是由心理学中的行为主义理论启发下的机器学习理论，强调的是在变化的环境中，不断适应环境的变化，从而获得最大限度的预期利益，利益的措施则是采取激励或惩罚的手段。在学习的过程中，学习者不需要关切具体执行的行为，而是通过最大利益的驱动使得模型付出具体行动。

深度强化学习在过去属于深度学习快速发展的一个重要领域。目的是利用计算机从决策的角度来辅助解决一般人工智能的问题。Google 的 DeepMind 团队已经将深度强化学习算法融入到视频、游戏、机器人等领域中，并取得了重要突破。2016 年，由 DeepMind 推出的 AlphaGo 计算机围棋程序，使用蒙特卡洛搜索和深度学习相结合的方式，使得 AlphaGo 的围棋水平甚

至超过顶级职业选手的水平，引起全球轰动。深度强化学习算法是 AlphaGo 的核心思想之一，使计算机通过自我博弈的方式不断提高棋力。深度强化学习算法可以基于深度神经网络实现从感知到决策控制的端到端学习，具有非常广泛的应用前景，其发展将进一步推动人工智能革命。

强化学习的理论具备普遍性，在诸如博弈论、运筹学、控制论、信息论、统计学和遗传算法等诸多学科中均具有研究价值。例如，在博弈论和经济学中，强化学习被用于分析在有限理性的条件下如何出现平衡问题；在控制论和运筹学中，强化学习则被当作“近似动态规划”。

在机器学习中，环境通常被规范为马尔可夫决策过程（Markov Decision Process，MDP），所以在这样的环境下，很多强化学习算法都可以利用动态规划技术的一些方法和技巧。强化学习与标准的有监督学习的不同之处在于，不会存在期望的输入与输出训练数据，也不会精准优化数据的权值。强化学习更专注于在线规划，探索未知的数据和现有知识之间的平衡，并调整现有知识结构以适应被探索的未知数据，并将探索过程利益最大化，传统的有监督学习方式更加注重模型最大限度地满足输入与输出的对应关系。此外，强化学习简单而言是一种基于试错方式的学习理论，没有直接的参数指导，只与环境相关，通过反复的试错得到最佳的运行策略。

12.1.2 基本原理

强化学习的基本逻辑原理如图 12-1 所示，核心思想在于如果代理的某个动作行为策略在环境中产生积极的影响，并获得环境的奖赏，则代理在该行为策略的趋势会逐步加强，反之则会逐步减弱，通过不断的行为策略，获得环境的奖赏以使得最终获得的奖赏最大化。

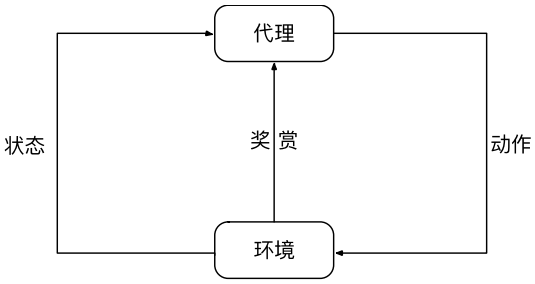


图 12-1 强化学习的基本逻辑原理

如图 12-1 所示，强化学习主要包括四个概念：代理、状态、动作、奖赏。代理（Agent）表示的是动作的发出者；状态表示代理所处的整体环境，可能是一个时间或位置；动作表示代理的行为；奖赏是用于衡量代理动作成功与否的反馈信息。

强化学习的过程是一种试探性学习过程，代理产生一个行为动作并作用于当前环境，环境接收到行为动作之后，状态发生变化，同时产生一个强化信号（奖赏）或弱化信号（惩罚）并告知代理，代理根据环境产生的强化信号或弱化信号选择下一个动作，选择下一个动作的原则

是强化信号则加强上一个动作，弱化信号则减弱上一个动作。选择的动作会实时影响环境的状态，环境也实时对代理做出反馈。

强化学习的训练过程是一个从无到有的过程，在初始状态，代理的动作是任意的，发出的动作也无规律可言，但是通过不断尝试，从错误中不断学习，最终找到规律，以达到学习的目的。强化学习的整个过程，无须人工标记的数据参与，极大减少了人工前期参与量，这也是强化学习备受关注的原因之一。强化学习与传统的有监督学习和无监督学习不太一样，强化学习更加注重最优决策。

深度强化学习算法被认为是智能体的“大脑”，而这个“大脑”至少包含两个模块：行为模块和评价模块。其中行为模块是智能体的执行机构，通过环境的外部状态，输出行为动作。评价模块可以认为是行为的评价者，评价模块会根据评价结果和环境反馈进行自我调节， 进而影响整个行为模块。这种智能体的行为—评价的模式与人类自身的行为存在一定的相似性。人类的行为也是在价值观和本能的指导下产生的，而且这些价值观不断会因为经验而不断改变或完善。在行为—评价的模式下，Google DeepMind 团队提出了一些深度强化学习算法，诸如 DQN、A3C 和 UNREAL 等，其中，UNREAL 被认为是目前最好的深度强化学习算法之一。

12.2　马尔科夫决策过程

12.2.1　马尔科夫过程

在了解隐马尔科夫模型之前，需要提前了解马尔科夫过程，可以将马尔科夫过程视为一个自动机，各个状态之间的转换存在一定的概率。若某个系统中，存在N个状态，在某个时间点t时刻都处于N个状态中的一个状态，就将这N个状态的集合视为$\{S_1, S_2, S_3 \dots, S_n\}$，利用$Q$表示在时间$1,2,3 \dots t$时的状态$\{Q_1, Q_2, Q_3 \dots, Q_t\}$。使用马尔科夫模型，则需要理解马尔科夫模型在使用之前的两点假设：

（1）当前状态的可能性只可能与前一个状态有关，与其他时刻无直接关系。

（2）状态的转移概率与时间无直接关系，时间只是一个状态表现系数。

也是基于假设 1，所以需要一个初始概率分布PI，其中$PI(S_n)$表示在最初时刻$t = 1$时当前状态为S_n的概率，这种概率与时间t无关。一个马尔科夫过程的示例如图 12-2 所示，即在不同时刻的状态以及状态之间的转移概率。

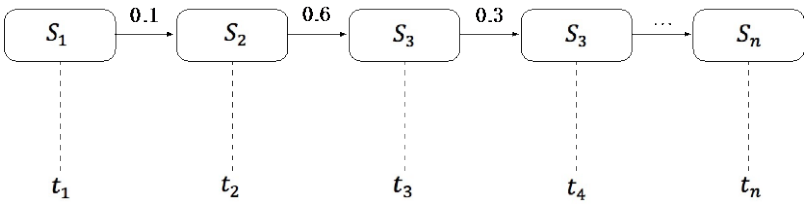


图 12-2　马尔科夫过程示例

12.2.2　隐马尔科夫模型

隐马尔科夫模型（Hide Markov Model）是一种概率统计模型，实质是一种隐藏状态的马尔科夫模型。在一般的马尔科夫模型中，状态对于观察者是可见的，但是在隐马尔科夫模型中，状态被隐藏，对观察者不可见，但是与状态相关的某些变量则是已知可获取的。

隐马尔科夫模型适合用于有未知条件的问题，它将隐藏的状态映射到N个可能的变量中，因此在某个时间t时，隐藏的状态会有N种可能。依次类推，以 1 为时间单位，则在$t+1$时隐藏状态也会有N种可能性，在$t+1$时刻的状态与时间t的状态是密切相关的，因此从时间t到时间$t+1$，隐藏状态有$N*N$种可能性。图 12-3 中的状态信息是N种可能变量，因此隐马尔科夫模型实际上包含两重随机过程，一方面状态之间的转移是一个随机过程；另一方面状态和可能的变量也是一个随机过程，它的过程如图 12-3 所示。

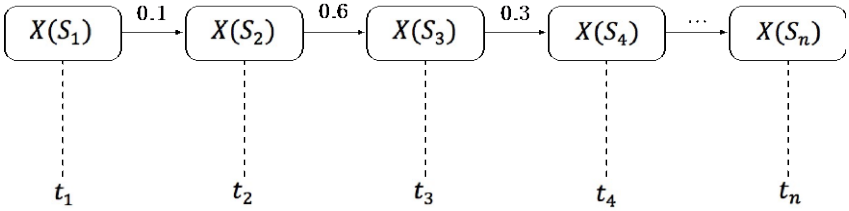


图 12-3　随机的马尔科夫过程示例

在图 12-3 中，将状态S替换为函数X，函数X表示在t时刻最大可能的状态，因此这里也不难看出隐马尔科夫模型的“隐”所表达的含义。在隐马尔科夫模型中，有五个非常重要的参数，下面结合医生给疑似感冒病人看病的过程解释这些参数。中医一般通过“望闻问切”的方法给病人看病，病人只有“正常”和“感冒”两种可能，但是无法直接给出疑似病人是“正常”还是“感冒”，需要对疑似病人进行观察，目的是推断疑似感冒病人是否真实感冒，如表 12-1 所示。

表 12-1　隐马尔科夫模型中的参数及其含义

参　　数	参数含义	示例含义
隐含状态S	序列中可能包含的隐藏状态	正常、感冒
观察状态O	可直接观察的变量	流鼻涕、打喷嚏、咳嗽
初始状态概率矩阵 ***PI***	隐藏状态的初始概率	疑似病人一般“正常”和“感冒”的概率
隐藏状态转移概率矩阵***A***	隐藏状态之间的转移概率	从“正常”到“感冒”的概率，从“感冒”到“正常”的概率
观察状态转移概率矩阵**B**	观察变量与隐藏之间的关联概率	观察状态是否“流鼻涕”导致“正常”到“感冒”的概率，“打喷嚏”导致“正常”到“感冒”的概率等

隐马尔科夫模型作为一种有监督的模型，需要通过语料库对模型进行训练，而训练过程中的主要任务是获得上述五个参数的值，求解相应的问题。使用隐马尔科夫模型可以求解三类常见的数学问题。

（1）已知模型的五个参数，求解某一特定序列的输出概率。此类问题一般是作为预测，预测可能所属的序列概率，在获得五个参数之后，可以采用 forward 算法进行后续求解。

（2）已知模型的五个参数，求解某一特定序列的隐藏状态序列。主要用于分析观察序列可能对应的隐藏状态。例如，利用观察状态“流鼻涕”“打喷嚏”“咳嗽”去分析疑似病人属于隐藏“正常”或“感冒”的概率，一般采用维特比算法进行后续求解。

（3）已知模型的最终输出序列，求解可能的状态转移概率，一般采用 Baum-Welch 算法以及 Reversed Viterbi 算法进行后续求解。

12.2.3　马尔科夫决策过程

在认识马尔科夫过程以及隐马尔科夫模型之后，认识马尔科夫决策过程则相对容易。马尔科夫决策过程提供了一个数学框架，用于在结果部分随机并部分由决策者控制的情况下对决策进行建模。马尔科夫决策过程可用于研究通过动态规划和强化学习解决的各种优化问题。马尔科夫决策过程最早在 1957 年就被大众知晓，广泛应用于自动化控制、经济学和制造等领域。

一个马尔科夫决策过程由一个六元组组成，四元组用M表示，$M = (S, D, A, \{P_{sa}(\cdot)\}, \gamma, R)$。$M$中的六个元素含义分别如下。

（1）S：表示所有状态的集合，对于任意一个状态$s_i \in S$，s_i表示第i步的状态。

（2）D：表示初始的状态分布，即S的概率分布。

（3）A：表示一系列动作的集合，对于任意一个动作$a_i \in A$，a_i表示第i步选择的动作。值得说明的是，对于动作的选择至少应该有两种。即$|A| \geqslant 2$。

（4）$P_{sa}(\cdot)$：表示在当前状态s的前提下，经过动作a之后，会转移到其他状态的概率分布情况。例如，在当前状态s_i的情况下执行了动作a_i，则转移到状态s'的概率为$p(s'|s_i, a_i)$。

（5）γ：被称作折扣因子，为一个[0,1]之间的实数。

（6）R：表示回馈函数。例如，在当前状态s_i的情况下执行了动作a_i，且转移到下一个状态s'的回馈函数可以记作$R(s'|s_i, a_i)$，倘若s'是状态s_i的情况下执行动作a_i的唯一的下一个状态，则可以简化为$R(s_i, a_i)$，$R(s_i)$表示当前状态下的回馈值。

在不考虑初始状态概率分布时，马尔科夫决策过程可以由五个元素组成，即$\mathrm{M} = (S, A, \{P_{sa}(\cdot)\}, \gamma, R)$。

根据初始分布D，可以得到某个代理的初始状态为s_0，然后从动作集合A中选择动作a_0来行执行，代理根据状态概率情况选择转移到另外一个状态s_1，然后不断重复上述过程，最终得到完整的状态转移过程，如图 12-4 所示。

s_0

图 12-4　状态与动作的转移示例

在马尔科夫决策过程中，事件是按照上述步骤一步步推进的，直到获得状态的序列s_0、s_1、s_2……s_n。最终对整个状态序列的折扣回馈进行求和，求和公式如下：

$$\mathrm{SUM} = \sum_{i=0}^{n} \gamma^i R(s_i)$$

由于γ的值一般情况下小于 1，因此序列中越远的状态得到的回馈越小。在经济学中，γ被称作无风险折现率，表示立刻能得到的回馈比未来的回馈更具价值。倘若γ的值趋近于 1，则可以将上述问题近似看作一个非折扣问题。在采用强化学习时，状态s_i的产生，伴随着相应的动作行为，强化学习的目标则是通过一系列动作行为使得上述公式最终的值足够大。

以蚂蚁走迷宫问题为例，如图 12-5 所示，以左上角(0,0)为起始点，以右下角(4,4)为结束点，蚂蚁需要从 5×5 的矩阵方格中从开始走到结束。

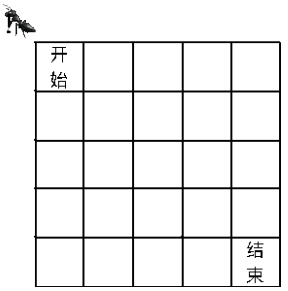


图 12-5　蚂蚁走迷宫的 5×5 矩阵示例

对于蚂蚁而言，5×5 的矩阵中的网格组成了马尔科夫决策过程中的状态集合S，而蚂蚁的动作集合A则包括了蚂蚁移动的四个方向，即向上、向下、向左、向右。由于蚂蚁并不是随心所欲地在矩阵中行走，会有一定的概率随机走到其他的邻近网格中，这些转移的概率即可编码为状态转移概率$P_{sa}(\cdot)$。例如，$p_{(0,0),左}(0,1)=0.7$，表示从矩阵中的(0,0)向左走到矩阵(0,1)的概率为 0.7。初始状态D则默认是从(0,0)开始，因此初始状态处于(0,0)位置的概率为 1，其余为 0。

整体而言，增强学习是在寻找一种最优策略π，使得从状态s中能够更好地从动作集合A中选择相应动作，对于每一个状态s，都通过最优策略π选择了动作，则预计会得到一个最大的回馈，公式表达为$E_{\pi}[\sum_{i=0}^{n}\gamma^{i}R(s_i)]$，其中，$E_{\pi}$表示所有状态之间的所有动作选择都是经过最优策略π。对于回馈函数r，是进行路径选择的关键一步，暗藏了最优化的目标方向，回馈函数的选择相对比较自由。例如，距离目标越近时给其加分奖赏，距离目标较远时进行扣分惩罚，且距离越远惩罚越重，越靠近目标点奖赏越高。

对于回报期望的总和可以通过价值函数（Value Function）进行表达，设定从状态s开始的价值函数公式如下：

$$V^{\pi}(s)=E_{\pi}[R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2R(s_2,a_2)+\cdots|s_0=s]$$

π依然表示策略，在价值函数的基础之上，可以定义最优价值函数，公式表达如下：

$$V^{*}(s)=\max_{\pi}V^{\pi}(s)$$

与价值函数类似的还有一个Q函数，Q函数表示对于一个确定的初始的状态，采取一个指定的动作，然后根据策略π采取后续动作，给出回报期望，公式定义如下：

$$Q^{\pi}(s,a)=E_{\pi}[R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2R(s_2,a_2)+\cdots|s_0=s,a_0=a\ \forall t>0\ a_t=\pi(s_t)]$$

同样的，在Q函数基础之上，也存在最优Q函数，公式表示如下：

$$Q^{*}(s,a)=\max_{\pi}Q^{\pi}(s,a)$$

对于价值函数和最优价值函数依然满足如下两个方程：

$$V^*(s) = \max_a R(s,a) + \gamma \sum_{s' \in S} P_{sa}(s') V^*(s')$$

$$V^{\pi}(s) = R(s,\pi(s)) + \gamma \sum_{s' \in S} P_{s\pi}(s') V^{\pi}(s')$$

上述方程为贝尔曼方程（Bellman Equation），也被称作动态规划方程，贝尔曼方程是动态规划能够达到最佳化的必要条件。上面的贝尔曼方程以递归的形式定义了V^*和V^{π}。$V^*(s)$表达的含义是采用的动作行为a，并且之后均采用最优的动作，求得所有期望回报的总和。$V^{\pi}(s)$表达的含义是如果持续采用策略π进行动作选择，则采取策略π的期望回报就是当前回馈加上未来的期望回报。

Q函数和价值函数是变量之间关系的表达形式，两者之间具备一定的关联性，$V^*(s)$与$Q^*(s,a)$的关系以及$V^{\pi}(s)$与$Q^{\pi}(s,a)$的关系如下公式所示：

$$V^*(s) = \max_{a \in A} Q^*(s,a)$$

$$V^{\pi}(s) = Q^{\pi}(s,\pi(s))$$

同理，将前面的贝尔曼方程改用Q函数表达，则可以得到：

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} P_{sa}(s') \max_{a' \in A} Q^*(s',a')$$

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P_{sa}(s') \max_{a' \in A} Q^*(s',\pi(s'))$$

对于多个马尔科夫决策过程，存在一个最优的策略π^*，使得从任意的状态s开始，π^*都能够获得最优的期望回报，π^*可以用下列两个等式进行定义：

$$\pi^*(s) = \arg\max_{a \in A} Q^*(s,a)$$

$$\pi^*(s) = \arg\max_{a \in A} R(s,a) + \gamma \sum_{s' \in S} P_{sa}(s') V^*(s')$$

根据上述公式，倘若已知Q^*，则可以比较容易地计算出最优策略，也可以从V^*以及P_{sa}中获得最优策略，但是相对复杂。价值函数和Q函数定义了最优策略，从上述公式可以看出，若需要得到最优策略，则需要尝试找到V^*或Q^*，一般可以通过价值迭代以及策略迭代的方式对问题求解。

（1）价值迭代。如果采用价值迭代的方式，则顾名思义需要更新每个状态s的价值$V(s)$，而且是在每次迭代中对每一个状态s都需要更新。由于会考虑每一个状态s的价值更新，即需要

考虑所有行为动作的可能性，因此价值迭代相对会比较耗时。

（2）策略迭代。策略迭代顾名思义是对策略的一种迭代方式，其收敛速度相对较快。

策略迭代和价值迭代的区别在于一个采用边试边学的方式，另外一个则是将所有情况考虑完整之后再做决定，可以根据实际情况选择策略迭代或价值迭代。

12.3　深度强化学习算法

12.3.1　DQN 算法

DQN（Deep Q Network）是谷歌 DeepMind 团队于 2013 年第一次提出的深度强化学习算法，并于 2015 年在 *Nature* 杂志发表，引起了行业内的广泛关注。DeepMind 团队将 DQN 应用于计算机游戏 Atari 中，完全的以人的视觉角度模拟人类玩 Atari 游戏，Atari 2600 总共包括 49 个独立的小游戏，经过 DQN 训练后，在 29 个游戏中打破了人类记录。DeepMind 团队基于 DQN 算法的程序取得了非常好的效果，并在某些领域超越了人类的水平，这是强化学习首次在业内提出并取得较好的效果。

DQN 是 Q learning 和神经网络的结合。Q learning 维护一张状态 State 到行动 Action 的得分表，记$Q(S_k, A_t)$为 State K 下执行 Action *t* 的得分，现在的问题是如何得到这张得分表。假设当前状态为S_1，S_1状态下可能的 Action 有A_1、A_2，S_1执行A_2后状态转移至S_2，获得的奖励为R，S_2的后续 Action 有A_3、A_4，记当前的$Q(S_1, A_2)$为$Q(S_1, A_2)_e$，称它为$Q(S_1, A_2)$估计，定义$\text{Max}Q(S_2) = \max(Q(S_2, A_3), Q(S_2, A_4))$，即$S_2$的候选状态中的最高得分，记$R + \gamma * \text{Max}Q(S_2)$为$Q(S_1, A_2)_r$，称为$Q(S_1, A_2)$现实。$\gamma$是未来对目前状态的影响衰减值，一般$0 < \gamma < 1$，$\delta = Q(S_1, A_2)_e - Q(S_1, A_2)_r$为现实和估计的差距，用$Q(S_1, A_2)_e + \alpha * \delta$去更新得分表中的$Q(S_1, A_2)$，$\alpha$是学习速率。选择动作可以有多种策略，其中一种叫作 Epsilon 贪心策略，加入 Epsilon=0.9，则有 90%概率会执行得分表中分值最高的 Action，10%的概率随机选择行为。Q learning 算法可描述为以下过程：

1. 初始化$Q(s, a)$、$\forall s \in S$、$a \in A(s)$为随机值，并且$Q\left(\text{终止}s, \cdot\right) = 0$
2. 重复（对每一节episode）：

    初始化状态S_t

    重复（对episode中的每一步）：

    使用同一个策略（如$Epsilon$贪心）跟进状态S选取一个动作执行

    执行完动作后，根据reward和最新的状态S_{t+1}更新

    $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha * \left(R_{t+1} + \text{Max}Q(S_{t+1}) - Q(S_t, A_t)\right)$$

$$S_t \leftarrow S_{t+1}$$

③ 循环直到S终止

在上面的介绍中，得分表$Q(s,a)$表在很多情况下会出现维度灾难，比如 Atari 游戏，输入是 210×160 像素的图片，每个像素点理论上有 256 种取值，那么状态就有$256^{210\times160}$种，计算机根本无法存储和计算。可以用一个函数来表示$Q(s,a)$，那么不管s的维度有多大，最后都可以通过矩阵运算得到Q。我们记$f(s,a,w)\approx Q(s,a)$，其中w为函数的参数集合。我们自然就会想到用深度神经网络来标识这个函数f。神经网络的输入就是所有的样本，将$R+\gamma*\mathrm{Max}Q(S_2)$即$Q(S_1,A_2)_r$作为前向计算的预期输出，那么 loss 函数就是$Q(S_1,A_2)_r$与$Q(S_1,A_2)_e$的误差，确定了损失函数后，就可以按照正常的神经网络的训练方法获得最终的$Q(s,a)$来表示函数f。

DQN 有一个 Experience Replay 策略，在每次 DQN 更新的时候，随机抽取一些之前的经历进行学习，这种做法打乱了经历之间的相关性，去除了样本分布的影响，达到了在记忆中学习的效果。DQN 的基本结构如图 12-6 所示。

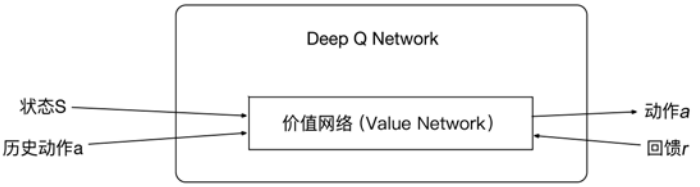


图 12-6　DQN 算法的基本结构

DQN 算法的输出是相对简单的离散输出，对于算法而言，最终的输出为有限的行为动作。在有限的行为动作输出前提下，DQN 算法仅仅使用一般强化学习算法中的评价模块，通过评价模块进行行为动作的选择。图 12-6 中的价值网络则是评价模块的体现，评价模块的输出为$Q(s,a)$，s表示状态，a表示动作，$Q(s,a)$表示在状态s个动作a选的价值。由于动作的行为是有限的，因此可以通过遍历某个状态下的各种动作的价值，然后选择其中最大的价值作为动作的输出。

与其他网络一样，需要通过梯度下降的方式对网络进行权值更新，使用梯度下降的前提是模型存在一个损失函数，因此对于图 12-6 中的价值网络则需要定义一个损失函数。在对 DQN 算法的实际使用中，价值网络可以根据具体解决的问题构建不同的网络形式。例如，当采用图像作为网络模型的输入时，则可以构建卷积神经网络作为价值网络。

DQN 算法是第一个公开的深度强化学习算法，但是模型的缺陷也很明显，DQN 算法仅仅使用了价值网络，必然导致模型的训练效率低。并且模型的输出为离散值，也必然导致模型的通用性不够。但是 DQN 算法毕竟是强化学习中最基本的算法模型，因此对于强化学习的发展

具有很强的开创性和突破性。

12.3.2　A3C 算法

A3C（Asynchronous Advantage Actor Critic） 算法是 DeepMind 团队于 2015 年在基于 DQN 算法的基础上提出的新算法，具有比 DQN 更好的通用性和计算效率。A3C 算法完全采用“动作—评价”（Actor Critic）框架，并引入了异步训练的方式，使得性能得到大大的提升。A3C 算法的核心思想在于“动作—评价”框架，即发出动作后，可通过评价模块评价动作的价值。如果动作的价值被认为是有益的，则调整动作模块加强该动作的可能性；反之，减少动作模块对该动作的可能性。通过反复训练，不断调整动作模块，最终找到模型的最佳动作。早期的 AlphaGo 的自学习方式，就是基于上述思想的。

基于“动作—评价”框架的基本思想，A3C 可以通过 DQN 方法更新评价模块的价值网络。一般强化学习中的动作模块输出有两种方式：一种是输出某种动作概率的输出；另一种是确定性的输出。对于 A3C 算法，则是使用概率输出的方式。如果评价模块对于动作的评价是积极的，则增加其概率，反之降低其概率。

不同于 DQN 算法的评价方式，为了获得更好的算法结果，A3C 算法在价值网络的输出$Q(s,a)$基础之上，新增了动作优势A作为另一个参考值。动作优势A表示当前动作相对于其他动作的优势。若状态s的价值为V，则动作优势$A = Q - V$。相对而言，采用动作优势的方式更容易被理解和接受。

A3C 算法为提高模型的训练速度采用了异步训练的思想，即同时在多个环境中进行训练。A3C 算法与 DQN 算法相比，采用异步训练大大提高了数据的采样速度，从而提高了训练速度。同时，利用多种培训环境收集样本，分布样本更均匀，更有利于神经网络的训练。同时采用完整的“动作—评价”框架，使得模型的结果更具准确性。

A3C 算法在 DQN 算法的基础上做出改进之后，使其在 Atari 游戏中的平均得分为 DQN 算法的 4 倍，在结果更好的同时，训练速度也大大提升，使得 A3C 算法替代了 DQN 算法。

12.3.3　UNREAL 算法

UNREAL（UNsupervised REinforcement and Auxiliary Learning）算法是 2016 年 11 月 DeepMind 团队提出的一种新的深度强化学习算法。UNREAL 是基于 A3C 算法的基础上提出的，但是在性能方面相对于 A3C 算法有很大地提升，在 Atari 游戏中是人类水平的 8.8 倍，在当时已经成为最好的深度强化学习算法，备受业内关注。

前面提到 A3C 算法之所以优于 DQN 算法，原因在于充分利用了“动作—评价”框架，该

框架的思想足够简单，以至于难以在基础思想架构中进行改进。 因此 UNREAL 算法基于 A3C 算法之上，通过在训练 A3C 算法的同时，并行训练多个辅助任务来改进算法。在 UNREAL 算法中，辅助任务主要包括两种类型，一种类型为控制类的任务，包括像素控制和隐层激活控制。像素控制是指输入图像变化的控制，使图像变化最大。因为图像的变化经常表明代理在执行非常重要的环节，通过控制图像的变化可以对动作的选择起到辅助作用。此外，隐层激活控制是控制隐藏层神经元的数量，目的是使激活量越多越好，激活的神经元越多，则其表达和拟合能力越强；另一类辅助任务是反馈预测任务，当神经网络具备反馈预测的能力时，则会使模型具有更好的表达能力。

UNREAL 算法通过辅助任务的形式，使得任务的准确度和效率都得到较好地提升。从另外一个角度而言，UNREAL 算法采用多任务的方式，对同一个目标进行训练，综合提升强化学习中动作模块的动作价值与优势。值得说明的是，UNREAL 算法并没有统改其他途径获得新的别的数据样本，而是在原有模型 A3C 基础之上的强化学习训练，因此其表达能力更强。

从 UNREAL 算法中可以学习到一个观点，即可以根据不同任务的特点针对性地进行辅助任务的设计，以改进算法。

强化学习经过近几年的发展， 在算法本身的效果上已经取得了越来越好的成就。从早期的 DQN 算法、A3C 算法、UNREAL 算法以及更多的改进算法，强化学习的效果已经逐渐超越人类。在未来的算法演进中，算法本身也将会继续被改进，强化学习的效果也将越来越引人瞩目。

12.4 强化学习的探索

12.4.1 应用场景探索

截至目前，强化学习的代表性里程碑是 Google DeepMind 团队的围棋程序 AlphaGo。AlphaGo 的围棋走子行为是强化学习中价值网络与策略网络共同强化的结果。据悉，Deep Mind 团队目前也在积极通过深度强化学习的方式训练计算机玩《星际争霸 2》游戏，并且极大可能会超越人类玩家的水平。

由于强化学习不需要人工标记的数据参与，因此可以极大地减少人工成本，应用前景应当非常广泛。阿里巴巴在双 11 的推荐场景中，采用自适应在线学习以及深度强化学习的方式，通过持续的模型优化建立推荐策略，对用户的行为与商品进行实时分析，帮助用户发现新的商品并推荐给用户，提高用户与商品的配对效率。日本的一家名为 Fanuc 的公司，通过工厂中的机器人在拿起一个物体时记录整个过程的视频，并记住每一次行为的结果是成功的或失败的，

通过模型调整，以使得下一次机器人可以更快更准确地拿起物体。对于金融投资决策也是如此，Pit.ai 公司采用强化学习的方式对交易策略进行评估，帮助用户建立交易策略，并帮助他们实现投资目标。

上述是目前已有的部分案例，总体而言，针对具备评价标准的应用场景都应该能够使用强化学习改进效率和提升效果。在未来，深度强化学习在以下几个方面很可能会被广泛应用。

（1）无人驾驶领域。强化学习的方式是通过自主的方式训练，价值网络可以更倾向于更安全的驾驶，通过一系列的动作使得驾驶过程更安全可靠。

（2）生物医疗领域。传统的人体免疫系统与强化学习的思想有一定的相似性，让强化学习参与到生物医疗领域中，为人类更有效地治疗某些疾病提供新的思路。

强化学习依然处于快速发展中，但显然强化学习的效果已经超出了预期，随着强化学习各方面的成熟，未来将会应用到更广阔的领域中。

12.4.2　面临的问题

虽然强化学习未来存在广阔的应用场景，但目前强化学习在实际落地中还面临如下一些问题。

（1）强化学习的自适应能力。强化学习本质上是一种基于试错方式的学习理论，当环境变化时强化学习需要通过反复试错，才能得到最佳的运行策略。然而，在无人驾驶这项应用中，多次试错的成本太高，因此，强化学习若要实际落地，就需要解决这样一个问题：如何在限制试错次数的情况下，获得正确的运行策略。

（2）强化学习的推理和想象力。强化学习是与环境相关的一种学习方法，学习的是人在环境中的行为。人类具有较强的主观能动性，例如，基于经验的推理能力以及丰富的想象力。如果强化学习也具备这些能力，那么就能够推理或想象在当前环境下做出某种动作带来的结果。如此，可以加快模型收敛速度。

（3）强化学习模型的可解释性。深度强化学习的价值函数和策略函数都是通过深度神经网络表示的，与生成对抗网络类似，这样的模型解释性都比较差。因此，若在强化学习的实际落地过程中遇到问题，则很难从原理和理论上给出解释，解决问题的难度也比较大。

强化学习在实际落地的进程中还存在不少问题亟待解决，期待广大读者能够投入此项研究，为深度强化学习的发展贡献力量。

12.5 本章小结

本章从深度强化学习的基本原理着手，逐步深入。并深入介绍了马尔科夫决策过程，其中包括马尔科夫过程以及隐马尔科夫模型，以使读者对强化学习的基础有深刻理解。随后介绍了Google Deep Mind 团队发表的典型强化学习算法，包括 DQN 算法、A3C 算法、UNREAL 算法，并对强化学习当前和未来可能的应用场景进行了介绍，也抛出了强化学习目前面临的问题。

深度强化学习是人工智能领域比较有潜力的研究领域，可以应对很多复杂的场景，从理论上也更接近强人工智能的目标。